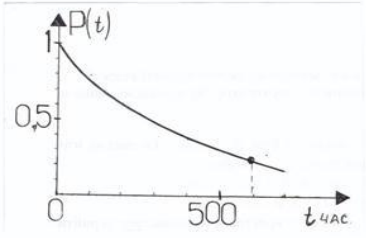
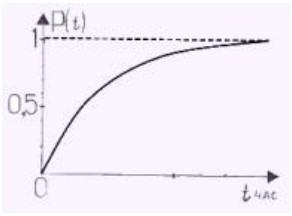
Функция надежности (вероятность безотказной работы) характеризуется производительностью на заданном промежутке времени.



$r(t) = e^{(-\lambda t)}$ λ – среднее число отказов появляющихся в машине в ед. времени.

$v = \frac{1}{\lambda} \rightarrow \lambda = \frac{1}{v}$; r(0) = 1; r(+∞) = 0 большие интервалы t

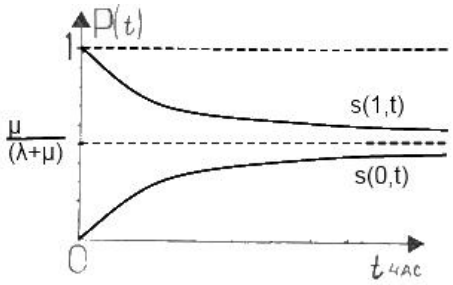
Функция восстановимости (вероятность восстановления работоспособного состояния) характеризует надежностные способности ЭВМ и восстанавливающего устройства одновременно(или дает информацию о том, как приспособлена машина к восстановлению своей производительности).



$u(t) = 1 - e^{(-\mu t)}$ μ - интенсивность восстановления $\tau = \frac{1}{\mu}$ среднее время восстановления работоспособности

$u(0) = 0$; $u(+\infty) = 1$ малые значения t 0.5;1;1.5

Функция готовности (вероятность того что в момент времени t>=0 работоспособна) одновременно учитывает и отказы и восстановления и характеризует ЭВМ в момент t>=0

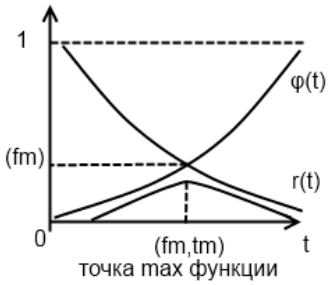


$s(0,t) = \frac{\mu}{\lambda+\mu} - \frac{\mu}{\lambda+\mu} * e^{-(\lambda+\mu)t}$ s(0,0) = 0;

$s(1,t) = \frac{\mu}{\lambda+\mu} + \frac{\lambda}{\lambda+\mu} * e^{-(\lambda+\mu)t}$ s(1,0) = 1;

$s = \frac{\mu}{\lambda+\mu}$ - коэффициент готовности 0 – ЭВМ отказала и восстановиться 1 – ЭВМ работоспособно

Функция осуществимости – характеризует качество функционирования ЭВМ относительно к процессу решения задач $f(t) = r(t)\varphi(t)$ **r(t)** – вер. безотказной работы φ(t) – вер. решения задачи на **n** работоспособных ЭВМ за время $\boldsymbol{t}$.



$\varphi(t) = \mathbf{1} - \mathbf{e}^{(-\boldsymbol{\beta t})}, \boldsymbol{\beta}$ – интенсивность решения задач на машине

Коэффициент накладных расходов

$\boldsymbol{\varepsilon} = \frac{\boldsymbol{tn}}{\boldsymbol{ty} + \boldsymbol{tc}}, \boldsymbol{tn} = \frac{\boldsymbol{l}}{\boldsymbol{v}}$;

$\boldsymbol{l}$ – разрядность, $\boldsymbol{v}$ – полоса пропускания

$\mathbf{t_n}$ **-** время передачи слова $\mathbf{t_y}$ **-** время операции умножения $\mathbf{t_c}$ **-** время операции сложения

Предыстория вычислительной техники

1.1 *Эволюция ВТ.*

Развитие человека и общества неразрывно связано с развитием техники вообще и вычислительной в частности, то есть человек постоянно стремится к созданию средств, повышающих его физические и вычислительные возможности. Для техники можно указать 2 ряда ветка развития

1) Физический

Рычаг (простейший механизм) → машины → конвейер (системы машин)

2) Вычислительный

Простейшие приборы → механические и электромеханические машины → ЭВМ → вычислительные сети → вычислительные системы

В истории вычислительной техники ясно выделяются 2 периода:

1) Древняя история

- Простейшие приборы (механические и электромеханические машины)
- Абак (Abacus)
- Логарифмическая линейка
- Арифмометр

2) Новая история

- ЭВМ,
- Вычислительные сети
- ВС

Простейшие вычислительные инструменты.

- **Абак** –

прибор дискретного действия. Счетная доска. 754-753г до н.э. Доска, разделенная на полосы, по которым передвигаются марки.

- **Логарифмическая линейка** –

была разработана в 17веке после создания учения о логарифмах. Автор Непер. Замена операций над логарифмами чисел. Носитель информации — отрезок прямой с логарифмической шкалой.

Абак и логарифмическая линейка свидетельствуют о дуализме вычислительной техники на самых ранних этапах.

Основные этапы развития цифровой вычислительной техники

- **Арифмометр** –

настольная машина механическая, с ручным или электрическим приводом. Является системой счетных колес. Дальнейшим продолжением арифмометра стали счетно-арифметические машины, которые использовались в начале XX века, для выполнения банковских и финансовых расчетах. Эффективно использовалась в механизации. Программу вычислений реализовывал сам человек.

1641 г. - был предложен арифмометр, Паскалем

1874 г. - Одне создал арифмометр

1890 г. - Серийное производство арифмометра из системы счетных колес

- **Машина Беббиджа.**

Была разработана профессором из Кэмбриджа в 1833 году. Представляет из себя универсальную цифровую механическую машину(Great Calculation Engine). По замыслу функциональная структура включала арифметическое устройство, память, средства ввода-вывода информации и управляющее устройство. Устройство управления работало на основе программы, то есть последовательности инструкций, записанных на перфокарту. В

перфокарте было 1050 чисел. По сути, это была первая машина, в которой была заложена возможность автоматизации вычислений. Это значит, что процесс вычисления мог автоматически изменяться в зависимости от полученного результата

5. Машина Цузе.

Машина Цузе — первая идея создания механического мозга (1935 г.). Было разработано Z семейство машин.

Z1 — 1938 г. - форма представления чисел двоичная, в качестве элемента переключателя используется механическая пластинка.

Z2 — 1940 г. - использование электромагнитных реле, но память оставалась механической.

Z3 — 1941 г. - двоичная электромеханическая машина с программным управлением.

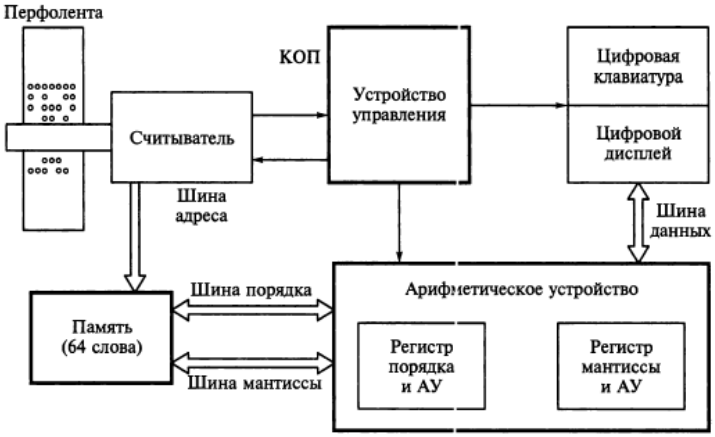


Рис. 1.1. Функциональная структура машины Z3:
→ — управляющие линии; КОП — код операции

АУ — арифметическое устройство
ЦК — цифровая клавиатура
ЦД — цифровой дисплей
П — память
Большие стрелки – операторы и операнды
Маленькие – управляющие сигналы

УУ — вырабатывало последовательность микроинструкций, каждая из которых соответствовала своей арифметической операции.

Перфолента хранила программы вычислений и данные для ввода информации использовались ЦК, а для воспроизведения ЦД, содержимое воспринималось с считывателя. Эти компоненты были реализованы как электромеханические.

Быстродействие 3-4 опер/с. , умножение 4-5 сек., в машине использовалось 2600 реле (600 АУ, 1400 П, 600 УУ), высота 6 фунтов, использовалась с расчетом создания ракеты

Анализ механических и электромеханических вычислительных машин

Арифмометры и счетно-аналитические машины развивались до 50-х годов XX столетия. Они полностью исчерпали возможности механики (механической «элементной» базы) и показали низкую эффективность ручного управления.

Следующим шагом в совершенствовании вычислительной техники явилось создание ВМ с программным управлением, использующих в качестве «носителя» программ перфокарты и перфоленты. В конце 1930-х годов начались работы по созданию ВМ на электромагнитных реле, т. е. внедрению электромеханической элементной базы.

1.2 ENIAC (Electronic Numerial Integrater Ard Computer)

Первая параллельная машина на физическом уровне!

Создана в электротехнической школе Мурра Пенсильваньского университета в США. Разработки велись в 1943-1945, но первые расчеты удалось выполнить только в 1946 году. Машина подвергалось неоднократной модификации и развитию. Находилась в эксплуатации до 1955 года. Руководителем разработки и идея принадлежит Мочли (Mouchly). Техническая реализация осуществлялась совместно с Эккертом (Eckert)

Принцип построения ENIAC

Машина ENIAC это вручную перестраиваемая конфигурация, состоявшая из трех подсистем: управляющей, собственно вычислительной и ввода-вывод.

УП – управляющая подсистема

ГПУ – главное программное управление

2хДПУ – дополнительное программное управление

ВП – вычислительная подсистема

20хУНС – устройство накопления и суммирования

УУМ – устройство умножения

УДК – устройство деления и извлечения квадратного корня

3хУХТ – устройство хранения таблиц

ПВв/Выв – подсистема ввода-вывода

УВв – устройство ввода

ВЗУ – внешнее запоминающее устойство

Устройство программного управления формирует управляющие сигналы для остальных устройств. Каждое из УНС могло суммировать и сохранять десятиразрядные числа в десятичной системе. В данной машине каждое вычислительное устройство (УНС, УУМ, УДК) разработано как специализированное арифметическое устройство и использовалось также, как и ячейки для хранения информации.

УХТ рассчитано на хранение 104 слов. В этой машине использовалась своеобразная система представления чисел. Для представления цифр использовалось 10 триггеров. Каждый триггер был реализован на одной лампе двойного триода. Каждый из 10 триггеров предназначен для одной цифры.

В машине параллелизм реализован на макро и микроуровнях: не все ВУ могли работать одновременно и параллельно обрабатывать разряды чисел. В машине не было отдельного запоминающего устройства. Для запоминания использовались УНС, УДК и УХТ. Суммарный объем составлял 334 слова. Функциональная структура позволяла подключать ВЗУ. Для решения задач программировались связи между элементами. Алгоритм функционирования не был проблемно-ориентирован и фиксирован, и требовалась ручная реконфигурация. Установка соединений между устройствами устанавливалась с помощью кабелей и штекеров.

Архитектурные возможности и Состав машины ENIAC

По своей архитектуре эта машина кардинально отличалась не только от предшествующих, но и от многих последующих машин. Параметры машины: частота 100 КГц, быстродействие 5000 операций сложения в секунду, 350 – умножения над десятиразрядными десятичными числами. Потребляемая мощность – 150 КВатт, вес – 27 тонн. Занимаемая площадь – 200 квадратных метров. Цена – 0,5 млн долларов. В ней было 18 тысяч ламп и 1500 реле. Машина ENIAC – вручную перестраиваемая конфигурация, состоящая из трех подсистем: управления, вычисления и ввода-вывода.

Анализ машины ENIAC

Достоинства:

1. Архитектура SIMD (по сути, машина ENIAC опередила элементную базу своего времени)
2. Параллелизм при обработке данных (допускалась параллельная работа ВУ и параллельная обработка десятичных разрядов чисел)
3. Реконфигурируемость структуры (ручное программирование не специализированной машиной под структуру решаемой задачи)
4. Однородность и модульность

Недостатки:

1. Ручное механическое трудоемкое программирование машины под структуру решаемой задачи.
2. Низкая надежность машины, обусловленная большим количествам ламп и реле.
3. Малая емкость ОЗУ, всего 334 числа.
4. Громоздкость и дороговизна.
5. Аппаратурная избыточность

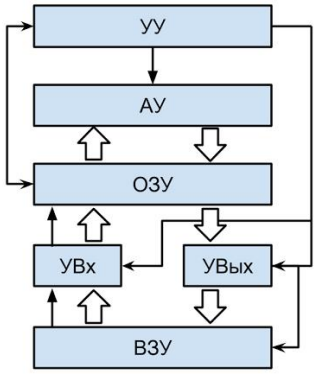
1.3 EDVAC

Разработана и построена в электротехнической школе Мурра, штат Пенсильвания, США. Разработки велись в 1944-1950 гг. В основу данной машины заложены принципы и архитектурные решения Мочли, Эккерта и Неймана. Машина была двоичной и последовательной.

Предпосылки создания ЭВМ с хранимой программой

Для решения актуальных задач требовались принципиально новые средства, то есть с автоматическим управлением вычислительным процессом, а это требует, чтобы машина могла хранить программу вычислений в своей памяти.

Функциональная структура и принцип построения машины EDVAC



Большие стрелки – операторы и операнды
Маленькие – управляющие сигналы
АУ – арифметическое устройство

АУ предназначено для выполнения арифметических операций, двоично-десятичного преобразования, операций пересылки из одного регистра в другой, а также из регистра в ОЗУ и обратно. Также АУ предназначено для операции выбора одного из двух чисел в зависимости от знака третьего. Регистры в АУ реализованы на линии задержки. В одном регистре могло храниться 32-разрядное слово.

ОЗУ состояло из 256 линий задержки, каждая из которых могла хранить 32 32-разрядных слова. Помимо линий задержки, в ОЗУ включены были переключательные схемы для взаимодействия с другими устройствами.

В качестве ВЗУ могли быть применены любые устройства со следующими носителями информации:

1) Перфокарты
2) Перфоленты
3) Магнитная лента
4) Фотопленка

ВЗУ работало в десятичной системе счисления

Увх использовалось для пересылки информации из ВЗУ в ОЗУ. Увых – обратно.

УУ использовалось для координации работы всех устройств при выполнении операций. Для каждой операции оно вырабатывало свою последовательность управляющих сигналов для остальных устройств.

В ячейках памяти хранится следующая информация:

a команда (например, +) b – первое слагаемое
a+1 второе слагаемое
a+2 следующая команда

Результат сохраняется в регистрах АУ

Последовательные действия ЭВМ в данном случае

1) Пересылка из ячейки a команды в УУ
2) Передача слагаемого из a+1 в АУ

3) Выполнение операции сложения
4) Запись результата в ОЗУ
5) Извлечение следующей команды из a+2 в УУ

1-ый бит определял: команда это или число → Если команда, то следующие 8 бит определяли код операции → Следующие 13 бит - адрес для сохранения результатов

Наряду с командами условного перехода в EDVAC реализована команда безусловного перехода. При выполнении этой команды следующая команда будет взята из ячейки в заданном адресе

| 1 | б. п. | адрес с |

Таким образом, EDVAC является полностью автоматическим и программируемым средством обработки информации

Анализ машины EDVAC

Технические характеристики:

Тактовая частота – 1 МГц. Быстродействие – тысяча операций в секунду, разрядность слова – 32 бита. Емкость памяти – 32768 байт

Данная машина имела фиксированную структуру, относилась к классу SISD.

Особенности:

1. автоматизация вычислений (возможность хранения программы в памяти и ее автоматической модификации)
2. Последовательный способ обработки информации
3. Фиксированная структуры (невозможность даже вручную реконфигурировать ЭВМ, за исключением ВЗУ)
4. Конструктивная неоднородность (все устройства логически и технически разные) Реализация этих принципов позволила создать технико-экономически эффективную ЭВМ для середины прошлого века

1.4 Пути развития отечественных средств ВТ(советую подготовиться к ответу по книги)

Этот путь тернист и сопровождался невосприятием кибернетики в 50-х годах и параллельных вычислительных технологий в 60-70-х годах. Тем не менее, для важнейших отраслей, ЭВМ в нашем государстве создавались и развивались.

МЭСМ (Малая электронная счетная машина). Первая ЭВМ в СССР и континентальной Европе.

Характеристики

Тактовая частота – 5 КГц, 50 операций в секунду над 17-разрядными числами. Память состояла из 94 слов, 63 для команд и 31 для данных. Двоичная система счисления. При конструировании использовалось 6 тысяч ламп, потребляемая мощность 25 КВт, занимаемая площадь 60 кв. м. Принципы построения были выведены Лебедевым независимо от Неймана. Команда условного перехода реализована программно.

БЭСМ 1 и 2 (Быстродействующая ЭСМ). Это была самая быстродействующая машина в Европе на тот момент – 10 тысяч операций в секунду. Построена на 5 тысячах лампах. Была достаточно надежной. Среднее время полезной работы – 72%. Мощность – 30 КВт. Площадь – 100 кв. м. Оперативная память реализована на ртутных линиях задержки.

М-20. В ней реализованы «зачатки» параллелизма: работа АУ при выполнении операции совмещалась с выборкой команд, а также с операциями ввода-вывода информации. Для этой машины впервые была создана система программирования ALGOL.

БЭСМ-4 являлась фактически модернизированным вариантом ЭВМ М-20. В ней были использованы полупроводниковые элементы и расширенная система команд.

Машина ЭВМ 5Э926. В этой ЭВМ впервые был реализован принцип многопроцессорности, внедрены новые методы управления внешними запоминающими устройствами, позволившие осуществлять одновременную работу нескольких машин на единую внешнюю память. Предусматривалась возможность объединения ЭВМ в системы.

Машина 5Э51. Это модифицированный вариант ЭВМ 5Э926.

БЭСМ-6. Была самой распространенной моделью семейства БЭСМ

Архитектурные достоинства:

• локальный параллелизм (на основе асинхронной конвейерной структуры);

• совмещение выполнения операций обращения к оперативной памяти с работой устройства управления и арифметико-логического устройства;

• совмещенный со счетом параллельный обмен массивами данных по шести каналам с магнитными дисками, барабанами и лентами;

• использование виртуальной памяти;

• возможность организации магазинного (стекового) способа обращения к памяти;

БЭСМ-6 поддержана схемами прерывания ЭВМ и защиты ее памяти, средствами преобразования математических адресов в физические оперативной памяти.

ЭВМ и вычислительные системы с программируемой структурой

1.5 Современный уровень ВТ

Архитектурные возможности обработки информации наращиваются главным образом за счет параллелизма и новых функциональных решений, а также за счет улучшения характеристик элементной базы.

В 1947 г. был создан первый транзистор. Автор – Шекин, Bells lab

1959 г. – создание первой интегральной схемы (чип) (Р. Нойс, Intel)

1961 г. – первая четырехтранзисторная схема

1970 г. – первый четырехразрядный чип (Intel 4004)

После 70-го года элементная база улучшалась в 2 раза каждые полгода. Чипы уже называли БИС

Современный уровень БИС

Тактовая частота – до 10^10 Гц. Кол-во транзисторов на кристалле – порядка миллиарда. Площадь кристалла – до 100 кв. мм. Размер подложки – до 500 кв. мм. Кол-во выводов – до 500. Потребляемая мощность – ватты. Технологические нормы – 22 нм. Предпринимаются попытки по созданию процессора с тактовой частотой 1 ТГц, но стабильности пока не добились. Уже сейчас очевидно, что будущее связано с созданием кристаллов, содержащих множество процессоров, т. е. параллельных ВС (System on Chip).

Прогресс

Первое направление – нанотехнологии
Второе – отказ от кремниевой технологии и создание какой-либо другой
Третье – квантовые компьютеры (основанные на законах квантовой механики)

2001 года – создан органический транзистор в Bells lab, первый в мире основанный на углероде, выращенный методом химической самосборки молекул. Длина канала транзистора – 1-2 нм.

Архитектура электронных вычислительных машин

2.1. Каноническая функциональная структура ЭВМ

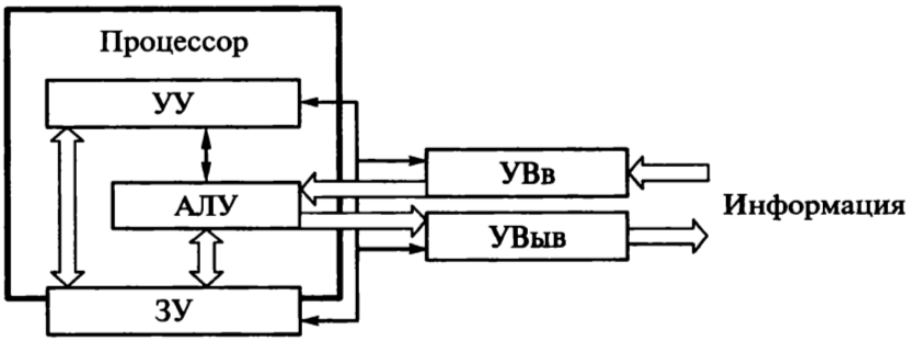
ЭВМ – средство, предназначенное для автоматической обработки информации (прежде всего для вычислительных и информационно-логических задач). Функционирование ЭВМ определяется не только ее технической реализацией (Hardware), но и также алгоритмом обработки данных. Этот алгоритм записывается на языке, доступном аппаратуре (программа). Программа в конечном счете интерпретируется на машинном языке. Как правило, это язык двоичных кодов. Говоря иначе, программа представляет собой последовательность команд, каждая из которых определяет операцию и данные, над которыми следует осуществить действие. Следовательно, в ЭВМ должны быть устройства для выполнения операций, а также для запоминания программ и данных. Функционирование ЭВМ не обходится без так называемых системных программ. Эти программы обеспечивают функциональную целостность ЭВМ, управление процессами и ресурсами, и выполняют функции сервиса.

ЭВМ – аппаратурно-программный комплекс, предназначенный для автоматического выполнения логико-вычислительной работы, ввода информации, ее обработки, хранения и вывода. Функциональная структура основывается на предложениях Джон фон Неймана.

Функциональное назначение ЭВМ

АЛУ служит для выполнения арифметических и логических операций над данными (операндами: числами или словами, в частности, буквенными последовательностями), а также операций условного и безусловного переходов; ЗУ используется для хранения программ и данных; УВв для ввода программ и данных, а УВыв для вывода из ЭВМ любой информации (в частности, результатов); УУ координирует работу всех остальных устройств при выполнении программ.

Структура ЭВМ Дж. Фон Неймана



АЛУ – арифметико-логическое устройство; ЗУ – запоминающее устройство; УВв – устройство ввода; УВыв – устройство вывода; УУ – устройство управления; Большие стрелки – операнды и команды; Маленькие стрелки – управляющие сигналы.

Архитектурные принципы построения ЭВМ:

- Программное управление работой ЭВМ
- Условный переход
- Принцип хранимой программы.
- Использование двоичной системы счислений для представления информации в ЭВМ.
- Иерархичность запоминающих устройств

Понятие о процессоре

Композицию из АЛУ, УУ и части ЗУ сейчас называют процессором. Если процессор имеет интегральное исполнение (т. е. представлен одной или несколькими БИС), то его называют микропроцессором. Главными характеристиками ЦПУ являются: тактовая частота, производительность, энергопотребление и архитектура.

Иерархия памяти

Иерархичность запоминающих устройств (ЗУ). С самого начала развития ЭВМ существовало несоответствие между быстродействиями АУ и оперативной памяти. Путем построения памяти на тех же элементах, что и АЛУ, удавалось частично разрешить это несоответствие, но такая память получалась слишком дорогой и требовала значительного количества электронных компонентов (что снижало надежность ЭВМ). Иерархическое построение ЗУ позволяет иметь быстродействующую оперативную память сравнительно небольшой емкости . При этом следующий более низкий уровень представляют внешние ЗУ на магнитных лентах, барабанах и дисках. Внешние ЗУ имеют относительно малую цену, обладают большой емкостью, но меньшим быстродействием, чем оперативная память. Иерархичность ЗУ в ЭВМ является важным компромиссом между емкостью, быстродействием, относительной дешевизной и надежностью.

2.2. Модель вычислителя

Вычислитель (Computer, Calculator) – тот, кто вычисляет что-либо или механический снаряд для вычислений (по Далю). В данном понятии заложен дуализм. Все вычислители основаны на примитивной имитации деятельности человека, занятого расчетами. Работа вычислителя не обходится без участия человека (оператора). Чем выше функциональные возможности вычислителя, тем реже происходит взаимодействие человека с вычислителем. Наивысшей степенью автоматизации обладают ЭВМ. Итак,

ЭВМ – результат технической интерпретации функциональной организации человека-вычислителя.

Следующая модель представляет основу функциональной организации ЭВМ – модель вычислителя

$$\mathbf{c = <h, a>}$$

h, a – описание конструкций и алгоритма вычислителя соответственно (h - конструкция вычислителя, a - алгоритм его работы)

Конструкция вычислителя может быть представлена как

$$\mathbf{h = <u, g>}$$

u – множество устройств

$$\mathbf{u=\{u_i\}}$$

g – описание структуры

Принципы, лежащие в основе конструкции вычислителя

Конструкция вычислителя основана на:

1) Последовательном выполнении операций (выполнение на множестве устройств u_i, соединенных в множестве g)
2) Фиксированной структуре (невозможность автоматического изменения g)
3) Неоднородности структуры и состава устройств

Алгоритм допускает представление в виде суперпозиции **a(P(D))**. Работа вычислителя определяется данными **D**, которые он обрабатывает, а также программой **P** для обработки этих данных. Алгоритм для заданных **P** и **D** должен приводить к однозначному результату. Таким образом модель вычислителя может быть записана, как **c = <u, g, a(P(D))>**. Машина EDVAC полностью основана на этих принципах, но с течением времени вычислительные машины отошли от них.

Понятие об аппаратурном и программном обеспечениях ЭВМ

Тенденция развития ЭВМ как аппаратурно-программного комплекса

Развитие ВТ по пути создания ЭВМ (как аппаратурных или аппаратурно-программных реализаций модели вычислителя или функциональной структуры машины дж. фон Неймана) может осуществляться в ограниченных пределах, обусловленных, в частности, конечностью скорости распространения сигналов в физических средах (кон, ^чностью скорости света, которая в вакууме равна (299 792 ± 0,4) км/с).

2.3. Понятие об архитектуре ЭВМ

Понятие архитектуры было впервые введено компанией IBM при создании семейства ЭВМ IBM-360 (1960-е годы). Под архитектурой понимается полная и детальная спецификация интерфейса “ЭВМ-пользователь”. Под пользователем следует понимать и программиста, работающего на ЭВМ, и инженера, и оператора, и любое техническое средство (терминал), т. е. все, что имеет доступ к ЭВМ, пусть даже через устройство ввода-вывода. Таким образом интерфейс – аппаратурно-программный посредник между ЭВМ и пользователем. При такой трактовке понятия архитектуры ЭВМ главным

становится то, что предлагается пользователю и как воспользоваться сервисом, предоставляемым со стороны машины.

Под архитектурой ЭВМ и любого средства обработки информации понимается совокупность свойств и характеристик, призванных удовлетворить потребности пользователя. Пользователя интересует, как правило, классы задач для решения на ЭВМ, возможность автоматизированного обучения программированию и языки программирования, возможности операционной системы (в частности, какие способы обработки информации доступны).

Среди характеристик или показателей пользователей интересует следующее: производительность, разрядность слов, оперативная память, характеристики устройств ввода-вывода, цена, а также показатель надежности, время автономной работы. Возможности, свойства и технические характеристики интересуют также и создателей (разработчиков). Они учитывают более широкий спектр свойств и характеристик. Таким образом под архитектурой ЭВМ следует понимать совокупность свойств и характеристик.

Определения понятия «архитектура вычислительного средства»

- Архитектура вычислительного средства - совокупность его свойств и характеристик.
- Архитектура вычислительного средства — концепция взаимосвязи и функционирования его аппаратурных (Hardware) и программных (Software) компонентов.

SISD-архитектура ЭВМ

Классификация архитектур на основе потоков команд и данных была предложена в 1966 г. профессором Стенфордского университета М.Д. Флиином. В соответствии с этой классификацией архитектура ЭВМ относится к классу SISD

Машина дж. фон Неймана имеет SISD-архитектуру. Архитектуру SISD и близкие к ней стали называть фоннеймановскими. Она предопределяет функциональную организацию ВМ, при которой состоит из двух основных частей: памяти, содержащей команды программы и данные, и процессора, выбирающего из памяти команды и их операнды и записывающего в нее результаты. Каждая команда явно или неявно указывает адреса операнда, результата и следующей команды.

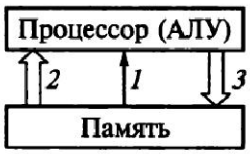


Рис. 2.2. SISD-архитектура ЭВМ:
1 — поток команд; *2* — потоки данных; *3* — потоки результатов

2.4. Понятие о семействе ЭВМ

Семейство – совокупность архитектурно близких ЭВМ, выделенных для фиксированного уровня развития вычислительной техники и электронной технологии. Границы семейства устанавливаются условно. Машины ЭВМ, принадлежащие одному

семейству, называют моделью. Как правило, название семейства связывают с названием фирмы-разработчика. Понятие семейства ЭВМ всегда ассоциируется с понятием совместимости. Под последним следует понимать, что программа, приготовленная для конкретной модели, дает один и тот же результат при ее исполнении на любой из модели семейства. При этом следует учитывать направление совместимости: сверху вниз или снизу вверх. Также совместимость ЭВМ в границах семейства проявляется в аппаратурном, программном и информационном планах.

Принципы построения семейств

Аппаратурная совместимость обеспечивается единством конструктивных решений, модульностью построения ЭВМ и стандартизацией связи и процедур управления.

Программная совместимость достигается единством функциональной структуры ЭВМ, единством команд и системой адресации. Набор команд любой из моделей обязательно принадлежит единой системе команд.

Информационная совместимость обеспечивается единым способом представления данных, единым способом представления файлов и использованием одинаковых носителей данных

Совместимость снизу вверх означает, что программа, написанная для младшей модели, может быть исполнена на любой другой старшей модели

Совместимость сверху вниз означает, что на старшей модели могут быть написаны программы для реализации на любой из младших моделей.

Примеры отечественных и зарубежных семейств

Примеры семейств – БЭСМ, Минск, Урал, ЕС ЭВМ, IBM-360, HP, DEC…

2.5. Поколение ЭВМ

Развитие средств обработки информации характеризуется сменой одного поколения другим, архитектурно более совершенным

Первые три поколения относятся к ЭВМ

В последние годы развитие ЭВМ осуществляется в основном за счет совершенствования элементной базы

Архитектура современных ЭВМ определяется возможностями микропроцессоров. На макроуровне современные ЭВМ ничем не отличаются по функциональной структуре от ЭВМ третьего поколения. Однако на микроуровне наблюдается кардинальный отход от концепции Неймана. Бурное развитие ЭВМ не позволяет указать сколь-нибудь длинный период, характеризующийся постоянством технических характеристик. Поэтому сейчас говорят о генерации ЭВМ в пределах семейства, но не присваивают номер.

Архитектурные возможности и показатели эффективности ЭВМ 1-го, 2-го и 3-го поколения

ω – показатель производительности (среднее число операции, выполняемое процессором при работе с оперативной памятью) [опер/сек]

V – объем оперативной памяти [бит]

Θ – среднее время безотказной работы ЭВМ [час]

σ – цена операций [цена ЭВМ/производительность

N	Годы появления	Характер использования	**a(P(D))**	Элементная база	Характер производства	ПО	Увв/выв
1	1949-1951	Одна задача в пассивном режиме	Последовательный, фиксированная структура	Лампы	Индивидуальный	Машинные языки, контрольно-наладочные тесты	Классические (дисплей, клавиатура, ВЗУ)
2	1955-1960	Набор задач, пассивный режим	Последовательно-параллельный, фиксированная структура	Полупроводниковые приборы	Мелкосерийное	Языки программирования, диспетчеры	Классические + каналы (для дистанционного управления машиной)
3	1963-1965	Мультипрограммная, активный режим	Последовательно-параллельное, ручное изменение структуры	Интегральные схемы	Серийное	Системы языков, операционные системы	Классические, каналы, оптические устройства

N	**ω**	**V**	**Θ**	**σ**
1	10^5	10^6	10^(0-1)	10^1
2	10^6	10^7	10^2	10^0
3	10^7	10^8	10^3	10^(-1)

1-е поколение

Фиксированность - функциональная структура раз и навсегда задавалась, и невозможно было ее изменить. То же самое относится к алгоритму программы

Индивидуальный характер производства - одна машина эксплуатировалась в течение нескольких лет

2-е поколение

Параллельный алгоритм связан с совмещением операций ввода-вывода с операциями, выполняемыми процессором

Диспетчеры – системные программы, обеспечивающее управление ресурсами вычислительной машины

Каналы – дистанционное управление ЭВМ

3-е поколение

Мультипрограммирование – реализация нескольких программ одновременно за счет пакетной обработки и разделения времени. В случае пакетной обработки в ЭВМ вводился набор задач, и была возможность контролировать работу процессора. В режиме разделения времени процессорное время разбивалось на кванты в соответствии с некоторыми правилами. Каждый квант процессорного времени доступ к ЭВМ имел

конкретный пользователь. Таким образом, разделялось процессорное время между пользователями, при этом у каждого пользователя сказывалось впечатление, что именно он владеет всей ЭВМ. Такой режим повышал эффективность эксплуатации процессора.

Распределение стоимости между компонентами ЭВМ

Стоимость ПО

От поколения к поколению стоимость ПО возрастала

1) 60% - Процессор и оперативная память
   По 20% - на Увв/выв и ПО
2) Процессор и оперативная память – 30%
   Увв/выв – 30%
   ПО – 40%
3) Процессор и оперативная память – 5%
   Увв/выв – 15%
   ПО – 80%

Затраты труда на разработку ПО для ЭВМ 3-го поколения оцениваются в 5000 человеколет. Системное ПО стоило 50 млн долларов

2.6. Производительность ЭВМ

В качестве комплексного показателя, как правило, используют дробь, в числителе которой находится показатель, который нужно максимизировать, а в знаменателе – показатель, который нужно минимизировать (пример: быстродействие/цена). Этот показатель может быть приемлем даже для низкопроизводительного средства, но очень дешевого. Для оценки эффективность нужно использовать вектор показателей.

Понятие о производительности ЭВМ

Производительность ЭВМ (Computer Performance):

1) ЭВМ имеет широкое распространение, и чтобы подобрать требуемую для данной области ЭВМ, нужно уметь оценивать показатель эффективности
2) ЭВМ или ее упрощенные конфигурации являются основными функциональными элементами при формировании параллельных ВС
3) ЭВМ и параллельные ВС составляют вычислительные сети

Под производительностью ЭВМ понимается ее способность обрабатывать информацию. Отметим, что когда говорят о производительности ЭВМ, то имеют в виду и ее потенциальные возможности, а не реальные, которые связаны с аномалией.

Показатели производительности ЭВМ

Для оценки производительности ЭВМ используют несколько показателей:

- Тактовая частота

  Количество элементарных операций или тактов, выполняемых ЭВМ в единицу времени. Время тактов – время выполнения одной элементарной операции. В первых ЭВМ были микросекунды, сейчас – наносекунды.

- Количество однотипных операций, выполняемых в единицу времени

Т. к. различные операции имеют различную длительность, то используют среднее их количество, выполняемое в единицу времени (т. н. номинальное быстродействие, пиковая производительность или техническое быстродействие Nominal Speed, Peak Speed). С

математической точки зрения, под номинальным быстродействием понимается среднее количество операций, выполняемых ЭВМ (при равномерном их выборе) при работе только с оперативной памятью в единицу времени.

{x_1, x_2, …, x_n} – часть набора операций, требующих обращение только к оперативной памяти. Пусть τ_i – время выполнения $\mathbf{x_i}$ и операции выбираются с равной вероятностью.

p = 1/n

Тогда математическое ожидание выполнения одной операции равно

$$\frac{1}{n}\sum_{i=1}^{n}\tau_i$$

Тогда номинальное быстродействие

$$\omega' = \frac{n}{[\sum_{i=1}^{n}\tau_i]}$$

Понятно, что при реализации на ЭВМ выбор операций неравно вероятен

Пусть $P(x_i) = \rho_i$. Тогда математическое ожидание выполнения одной операции равно

$$\sum_{i=1}^{n}\rho_i\tau_i$$

Обратная ей величина – быстродействие по Гибсону. Коэффициент ρ_i также называют весовым

$$\omega^o = \frac{1}{[\sum_{i=1}^{n}\rho_i\tau_i]}$$

Существует несколько «смесей» Гибсона – наборы вероятностей, которые отображают статистику задач, решаемых на ЭВМ. На практике часто используют модифицированные значения ω' и ω^o, когда в набор операций включаются лишь операции с фиксированной запятой. Показатели производительности зависят не только от количества операций и вероятности их выбора, но и от затрат машинного времени на ввод программы и данных, обращения к внешней памяти, работы операционной системы, выдачи результатов и т. д. Поэтому для характеристики производительности ЭВМ при решении задач целесообразно ввести дополнительный показатель

- Пусть **{I_1, I_2, …, I_l}** – набор типовых задач, т. е. эталонных тестов или программ оценки производительности (Benchmark).

$\mathbf{V_i}$ – число операций, непосредственно входящих в программу решения $\mathbf{I_i}$

$\mathbf{t_i}$ – время решения $\mathbf{I_i}$ (сюда входят расчетное время и дополнительные затраты машинного времени)

Быстродействие ЭВМ при решении $\mathbf{I_i}$ – величина $\omega_i = \frac{v_i}{t_i}$

Отношение $\frac{1}{\omega_i}$ – среднее время выполнения одной операции при решений I_i

Пусть $\{\boldsymbol{\pi_1}, \dots, \boldsymbol{\pi_L}\}$ – распределение вероятностей спроса на решение I_i

Тогда

$$\sum_{i=1}^{L}\frac{\pi_i}{\omega_i}$$

– среднее время выполнения одной операции при решении набора типовых задач

Соответственно, среднее быстродействие ЭВМ есть величина, равная

$$\omega = \frac{1}{\sum_{i=1}^{L}\frac{\pi_i}{\omega_i}}$$

Существует множество тестовых наборов $\{\boldsymbol{I_i}\}$

Одни из популярных – LINPAC, NAS Parallel Benchmark. Он состоит из набора программ для решения задачи линейной алгебры и позволяет оценить производительность ЭВМ на вычисления с плавающей запятой.

- Среднеэффективное быстродействие

$\omega^* = \frac{1}{\sum_{i=1}^{L} \frac{\pi_i}{\omega_i^*}}$, $\omega_i^* = \frac{v_i^*}{t_i}$ где v_i^* – число унифицированных операций, с помощью которых можно интерпретировать программу решения I_i. В качестве унифицированной может быть взята операция, через которую могут быть выражены все остальные операции в каждой из сравниваемых несовместимых машин.

Единицы измерения производительности ЭВМ

Для оценки номинального быстродействия и быстродействия по Гибсону в случае, когда учитываются только операции с фиксированной запятой, применяется MIPS (Million Instructions Per Second) и GIPS (10^9). Измерение производительности на тестовых наборах задач осуществляется в FLOPS (FLoat-point Operations Per Second).

2.7. Показатели, характеризующие память ЭВМ

Количество информации (по К. Шеннону) структурные единицы информации

Для характеристики возможностей памяти используют ряд показателей. Все они связаны с количеством информации. Количество информации, как понятие, было введено в 1948-ом году К. Шенноном.

$$H = -\sum_{i=1}^{n} p_i \log_2 p_i$$

H – одно из устойчивых состояний памяти

p_i – вероятность нахождения памяти в состоянии i

Единица количества информации – бит – то количество информации, с помощью которого можно задать одно из двух равновероятных альтернативных состояний памяти. Слово «бит» также используют для определения разряда двоичного числа. Запоминающее устройство, способное хранить 1 бит информации, называется элементом или ячейкой памяти. Самый распространенный элемент памяти – триггер.

Для измерения количества информации используются следующие единицы

Единица	Бит B; t	Тетрада Tetrad	Байт Byte	Слово	Массив Array
H	1	4	8	l бит	L слов
Количество состояний N=2^H	2	16	256	2^l	2^(L*l)

Емкость памяти

Количество информации, которое может храниться в памяти. В качестве основных единиц выступают биты и байты. В качестве укрупненных единиц используются: Кбит (1024; 2^10), Мбит(2^20), Гбит (2^30), Тбит (2^40), Пбит (2^50)

Ширина и время выборки

Ширина выборки - определяется количеством информации, записываемой или считываемой из памяти за одно обращение

Время выборки – промежуток времени с момента подачи сигналов чтения или записи до завершения соответствующей операции

Время обращения к памяти складывается из времени выборки и времени подготовки памяти к следующему обращению. Также это время называют длительностью цикла обращения к памяти. В динамической памяти всегда требуется регенерация памяти, поэтому в ней время обращения больше времени выборки.

Быстродействие памяти

Быстродействие памяти характеризуется пропускной способностью или скоростью обмена информации между ней и другими устройствами. Определяется количеством информации, которая может быть записана в память или считана из нее в единицу времени. В качестве основной единицы используют

бод = boud = бит/с = bit/s

Кбод = 10^3 бод

Мбод = 10^6 бод

2.8. Надежность ЭВМ

Основные понятия и показатели надежности ЭВМ

Отказ – событие, заключающееся в том, что ЭВМ теряет способность выполнять возлагаемые на нее функции по обработке информации. Отказ может быть полный или частичный. В случае полного отказа ЭВМ теряет способность выполнять любую из возлагаемых на нее функций. Это событие носит постоянный характер, и оно само не устраняется. Имеют место сбои при работе ЭВМ – самоустраняемые события. Для устранения отказа принимаются специальные меры

Восстановление – событие, при котором отказавшая машина приобретает способность выполнять все возложенные на нее функции. Будем считать, что восстановление происходит с помощью специального восстанавливающего устройства. Его природа может быть самой различной. Это может быть действительно техническое средство, либо микропрограммное устройство, либо комплекс программ, либо даже бригада инженеров. Таким образом, восстановление может быть автоматическим и полуавтоматическим с участием людей.

$\Omega(\tau)$ – производительность в момент времени $\tau >= 0$

$$\Omega(\tau)=\begin{cases}1, \text{если ЭВМ в } \tau \geq 0 \text{ исправна} \\ 0, \text{в противном случае}\end{cases}$$

Вероятность безотказной работы и интенсивность отказов ЭВМ

ξ – случайная величина, являющаяся моментом первого отказа в работе ЭВМ. Главным показателем является функция надежности ЭВМ. Она характеризует производительность ЭВМ на промежутке времени

$$r(t) = \mathrm{P}\{\forall\tau \in [0, t) \rightarrow \Omega(\tau) = 1\}$$

Функция означает вероятность того, что $\forall\tau$ из промежутка $[0, t)$ производительность ЭВМ равна потенциально возможной. Она обладает набором свойств. Для их понимания дадим еще одно определение

$r(t) = \mathrm{P}\{\xi > t\}$, где $\mathrm{P}\{\xi > t\}$ – вероятность события, что в момент ξ возникновения первого отказа наступит после t >= 0

1) r(0) = 1 (Машина в начальный момент времени исправна)
2) r(+inf) = 0 где $\mathrm{P}\{\xi > (+inf)\} = 0$ (Машина исправна на конечном промежутке времени)
3) $r(t_1) >= r(t_2)$ при $t_1 <= t_2$ (Чем больше прошло времени, тем более вероятность того, что машина откажет)

Событие $\xi > t2$ и $t2 \geq \xi > t1$

По теореме сложения вероятностей

$$r(t) = \mathrm{P}\{\xi > t_1\} = \mathrm{P}\{(\xi > t_2) \cup (t_2 \geq \xi > t_1)\} = P\{\xi > t_2\} + \mathrm{P}\{t_2 \geq \xi > t_1\} \geq \mathrm{r}(t_2)$$

Также используется функция ненадежности ЭВМ

$$q(t) = 1 - r(t)$$

$$q(t) \sim \frac{n(t)}{N}$$

N – число исправных машин в начале испытания

n(t) – число отказавших машин к моменту времени t

q(t) используется для вычисления времени безотказной работы машины

V – среднее время безотказной работы

$\lambda(t)$ - интенсивность отказов

$$V = \int_0^\infty t dq(t) = -\int_0^\infty t dr(t) = -tr(t)\Big|_0^\infty + \int_0^\infty r(t)dt \sim \frac{1}{N}\sum_{i=1}^{N} t_i$$

$$\lambda(t) = \frac{1}{1 - q(t)} * \frac{dq(t)}{dt} = -\frac{1}{r(t)}\frac{dr(t)}{dt} \ (***)$$

$\lambda(t) = \frac{n(\Delta t)}{N(t) * \Delta t}$ *(**)*

Для доказательства (**) воспользуемся формулой $q(t) = \frac{n(t)}{N}$ $(*)$

$n(\Delta t) = n(t + \Delta t) - n(t) \sim N * [q(t + \Delta t) - q(t)]$

$N(t) = N - n(t) \sim N[1 - q(t)]$

Подставив найденные оценки в формулу (**) и перейдя к пределу при $\Delta t \to 0$, можно убедиться в справедливости формулы для оценки $\lambda(t)$

N(t) – число исправных машин в момент времени t

n(Δt) – число отказавших машин в промежутке времени [t, t+ Δt]

Проинтегрировав от 0 до t выражение (***), получим

$$\int_0^t \lambda(t)dt = -\ln r(t)$$

$$r(t) = \exp\left[-\int_0^t \lambda(t)dt\right]$$

$r(t) = e^{-\lambda t}$

$V = \int_0^{\infty} r(t)dt = \int_0^{\infty} e^{-\lambda t}dt = -\frac{1}{\lambda} * e^{-\lambda t}|_0^{\infty} = \frac{1}{\lambda}$

$\lambda = \frac{1}{V}$ [1/час] – среднее число отказов, которое может произойти в ЭВМ в единицу времени

Функция r(t) – вероятность того, что в ЭВМ не произойдет ни одного отказа за время t

Тогда вероятность появления k отказов за время t равна

$r_k(t) = \frac{(\lambda t)^k}{k!} * e^{-\lambda t}$

$r_0(t) = r(t)$

Найдем мат. ожидание того, что в ЭВМ за время t произойдет k отказов

$\sum_{k=1}^{\infty} k * r(t) = e^{-\lambda t} * \lambda t * \sum_{k=1}^{\infty} \frac{(\lambda t)^{k-1}}{(k-1)!} = \lambda t$

Где $\sum_{k=1}^{\infty} \frac{(\lambda t)^{k-1}}{(k-1)!} = e^{\lambda t}$

Т. о. $r_k(t) = \frac{(\lambda t)^k}{k!} * e^{-\lambda t}$

Закон распределения времени безотказной работы является экспоненциальным, а поток отказов в элементарной машине – пуассоновский (простейший)

Вероятность и интенсивность восстановления ЭВМ

Показатель, характеризующий надежностной способности и ЭВМ, и восстанавливающего устройства одновременно. Говоря иначе, эта количественная характеристика дает информацию о том, как приспособлена ЭВМ к восстановлению своей производительности после отказа с помощью восстанавливающего устройства.

$$u(t) = 1 - P\{\forall \tau \in [0, t) \to \omega(t) = 0\}$$

Где $P\{\forall \tau \in [0, t)\}$ – вероятность того, что при выполнении восстановительных работ в машине $\forall \tau \in [0, t)$ производительность $\omega(t)$ остается равной 0. Другими словами, это вероятность того, что отказавшая ЭВМ при работе восстанавливающего устройства не будет восстановлена за время t

1) u(0)=0
2) $u(+\infty) = 1$
3) $u(t1) \leq u(t2)$ при $t1 \leq t2$

Для практической оценки восстановления ЭВМ используется следующая формула:

$$u(t) \sim u^{\sim}(t) = \frac{m(t)}{M}$$

Где M – число отказавших машин в начале восстановления, а m(t) – число восстановленных машин за время t при условии, что ремонт каждой ЭВМ осуществляется своим восстанавливающим устройством

$$u(t) = 1 - e^{-\mu t}$$

Где μ – интенсивность восстановления ЭВМ

1/ μ – среднее время восстановления одной ЭВМ

Примечания:

1) Проведение статистических экспериментов, позволяющих вычислить $q^{\sim}(t), V^{\sim}, \lambda^{\sim}(t)$ для машин 1-3 поколений невозможно было осуществить. Только в 3-ем поколении с появлением микро и минимашин стало возможным проведение таких экспериментов для упрощенных конфигураций (например, процессор с памятью)
2) При отыскании вышеупомянутых оценок целесообразно использовать эргодическую гипотезу. Эта гипотеза позволяет вместо статистических результатов наблюдения за большим числом N машин воспользоваться результатами наблюдения за одной машиной в течение длительного промежутка времени
3) Справедливость экспоненциального закона для функции надежности r(t) подтверждена обработкой статистических данных при эксплуатации машин первых трех поколений. При проверке этой гипотезы был применен критерий согласия χ^2 Пирсона
4) Среднее время безотказной работы в современных микропроцессорных ЭВМ оценивается многими годами (V<=10^8 часов). Поэтому для определения оценок показателей надежности ЭВМ разработаны методики ускоренных экспериментов (например, использующие нагревание интегральных схем)

Функция и коэффициент готовности ЭВМ

Функция надежности ЭВМ характеризует производительность машины на промежутке времени [0, t). А функция восстановимости информирует о том, что ЭВМ на этом промежутке времени приобретает способность выполнять возложенные на нее функции. Соответственно, первое понятие связано с отказом, второе – с восстановлением. Это – характеристики начального периода работы ЭВМ. Они неинформативны при длительной эксплуатации ЭВМ.

$\lim_{t\to\infty} r(t) = 0$

$\lim_{t\to\infty} u(t) = 1$

Поэтому нужен комплексный показатель, который связан и с отказом, и с восстановлением, и который был бы информативен при переходном и стационарном режимах работы.

Введем следующее обозначение

$E = \{0, 1\}$ – пространство состояний ЭВМ

Если машина находится в состоянии 0, то принято считать, что она отказавшая. Если 1, то работоспособна и исправна.

$p_j(i, t)$ – вероятность нахождения ЭВМ в состоянии j в момент времени t при начальном состоянии i

Функцией готовности ЭВМ является следующее:

$S(i, t) = P_1(i, t) = P\{i, \omega(t) = 1\}$ – вероятность того, что ЭВМ, начиная функционировать в состоянии i в момент времени t, имеет потенциально возможную производительность

Требуется вывести дифференциальное уравнение для расчета функции готовности
Свойства функции готовности

1) S(0, 0)=0

S(1, 0)=1

Функция готовности зависит и от отказа, и от восстановления, и характеризует производительность ЭВМ в момент времени t>=0. Следовательно, в качестве начального состояния может быть взято одно из состояний: 0 или 1. Справедливость данного свойства вытекает из определения функции готовности

2) $\lim_{t\to\infty} S(i,t) = S = const \quad 0 < S < 1$

Функция готовности характеризует поведение машины в любой момент времени t, т. е. не только в переходном, но и стационарном режимах работы. Ясно, что в режиме длительной эксплуатации и при наличии процедуры восстановления можно сказать, что вероятность отказа не равна 1. Следовательно, данное свойство справедливо

3) $S(0, t_1) <= S(0, t_2)$

$S(1, t_1) >= S(1, t_2)$ при $t_1 <= t_2$

Справедливость этого свойства вытекает из первых двух

Величина S – коэффициент готовности. Он не зависит от начального состояния i. Это объясняется тем, что за длительное время происходят и отказы, и восстановления. Следовательно, забывается предыстория, то есть теряется зависимость от i.

Физический смысл функции готовности - вероятность того, что ЭВМ в момент времени t>=0 работоспособна, т. е. способна выполнять возложенные на нее функции. Следовательно, эта функция информирует о том, может ли пользователь начать работу на ЭВМ в данный момент времени. Если же ЭВМ общего назначения и находится в режиме длительной эксплуатации, то пользователь может оценить возможности работы на ней по коэффициенту готовности.

Для вывода уравнения получим предварительную оценку

Пусть Δt – промежуток времени бесконечно малой величины

Нужно найти S(i, t+ Δt)

Оценим вероятность того, что ЭВМ в момент времени t+Δt находится в работоспособном состоянии. Это событие сложное и может наступить, если произойдет одно из несовместимых событий:

1) ЭВМ в момент времени t исправна и за Δt не произойдет отказ

$P_1(i, t)=S(i, t)=r(\Delta t)$

Для оценки применим разложение в ряд Маклорена:

$$r(\Delta t) = r(0) + \frac{r'(0)}{1!} * \Delta t + \frac{r''(0)}{2!} * (\Delta t)^2 + \cdots = e^{-\lambda * \Delta t} = 1 - \lambda * \Delta t + O(\Delta t)$$

2) ЭВМ в момент времени t неисправна и за Δt будет восстановлена

$P_0(i, t)=1-S(i, t)=u(\Delta t)$

$$u(\Delta t) = u(0) + \frac{u'(0)}{1!} * \Delta t + \frac{u''(0)}{2!} * (\Delta t)^2 + \cdots = 1 - e^{-\mu * \Delta t} = 0 + \mu * \Delta t + O(\Delta t)$$

Заметим, что ЭВМ может оказаться в момент времени $t + \Delta t$ в работоспособном состояний в результате других сложных событий. Например, в момент времени t ЭВМ работоспособна, а за Δt происходит и отказ, и восстановление:

$r^1(\Delta t)*u(\Delta t)=\frac{\lambda*\Delta t}{1!}*e^{-\lambda*\Delta t}*[1-e^{-\mu*\Delta t}]=\lambda*\Delta t*[1-\lambda*\Delta t+O(\Delta t)]*[\mu*\Delta t+O(\Delta t)]=O(\Delta t)$

$S(i,\ t+\Delta t)=S(i,\ t)*r(\Delta t)+[1-S(i,\ t)]*u(\Delta t)+O(\Delta t)=S(i,\ t)*[\ 1-\lambda*\Delta t+O(\Delta t)]+[1-S(i,\ t)]*[\ \mu*\Delta t+O(\Delta t)]+\Delta t$

Перенесем в левую часть S(i, t), поделим обе части равенства на Δt и перейдем к пределу при $\Delta t\ \to 0$

$$\frac{d(S(i,t+\ \Delta t))}{\mathrm{dt}}=\mu=(\lambda+\mu)*S(i,t)$$

$$S(0,t)=\frac{\mu}{(\lambda+\mu)}-\frac{\mu}{(\lambda+\mu)}*e^{-(\lambda+\mu)*t}$$

$$S(1,t)=\frac{\mu}{(\lambda+\mu)}-\frac{\lambda}{(\lambda+\mu)}*e^{-(\lambda+\mu)*t}$$

Легко заметить, что $\lim_{t\to\infty}S(i,t)=\frac{\mu}{(\lambda+\mu)}=const=S$

Для современных микропроцессоров $\lambda\leq 10^{-8}\left(\frac{1}{\text{час}}\right)$, $\mu\leq 1\ \left(\frac{1}{\text{час}}\right)$

Замечания

1) Современные ЭВМ достаточно быстро входят в стационарный режим работы, поэтому на практике достаточно использовать коэффициент готовности
2) Довольно часто используют выражение «надежность=0.999». Это – жаргонное выражение, не имеющее однозначного толкования. Под надежностью здесь можно понимать значение или функции надежности, или функции готовности для некоторого времени, но наиболее вероятно полагать под этим значение коэффициента готовности

Функция осуществимости решения задач на ЭВМ

Цель функционирования ЭВМ – решение поступивших задач, т. е. выполнение программ. Введенные нами показатели устанавливают взаимосвязь между производительностью ЭВМ и ее надежностью. Ясно, что они характеризуют качество работы ЭВМ без относительно процесса решения задач. Этот пробел можно устранить, если использовать функцию осуществимости решения задач

$f(t)=r(t)*\varphi(t)$, где $\varphi(t)=P\{0<=\nu<t\}$

ν – момент решения задачи на полностью работоспособной ЭВМ

$\varphi(t)=1-e^{-\beta t}$, где β – интенсивность решения задач, 1/ β – среднее время решения задач

2.9. Предпосылки совершенствования архитектуры ЭВМ

Использование канонической функциональной структуры Неймана и модели вычислителя не позволяет в полной мере удовлетворить потребности общества. Требуются средства, которые не могут быть построены на принципе последовательной обработки

Эволюция структуры канонической ЭВМ Дж. Фон Неймана

Если исходить из глобального трактования архитектуры ЭВМ, то легко заметить, что первые три поколения ЭВМ полностью основываются на модели вычислителя, на принципах, положенных в ее основу. Архитектуры ЭВМ, принадлежащих второму и даже третьему поколениям, являются модификациями архитектуры дж. фон Неймана.

Анализ возможностей совершенствования ЭВМ

Оценим пределы, которые следуют из принципов модели вычислителя:

1) Последовательная обработка информации
   a) Совершенствование и разработка алгоритмов решения задач
   b) Создание эффективных систем программирования и оптимизации программ
   c) Повышение быстродействия элементной базы
   d) Улучшение алгоритмов выполнения машинных операций
   e) Модернизация алгоритмов управления вычислительными процессами и канонической структуры ЭВМ

Первые 2 способа основаны на фундаментальных достижениях математики и скрупулезного программирования, которое в максимальной степени учитывает архитектурные возможности ЭВМ. Эти способы следует применять для уникальных машин, ориентированных на решение специальных классов задач. Для универсальных машин возможности исчерпаны

Третий способ, то есть повышение быстродействия элементной базы в пределах кремниевой технологии близки к пределу (проблема теплоотвода, ограничение времени переключения 10^-11 сек - время переключения на тонких пленках уже сейчас составляет 10^-10 – 10^-11 сек. Если учесть скорость распространения сигналов в компьютере (а она не может превышать скорость распространения света в вакууме – 299794 +- 0,4 км/с) и принцип неопределенности Гейзенберга, то мы также выходим на эти ограничения по времени переключения) и дальнейший процесс связан с использованием углеродных молекул, на основе которых можно строить элементную базу

Закон Мура – эмпирическое наблюдение, согласно которому количество транзисторов, размещаемых на кристалле ИС удваивается каждые 2 года. Часто цитируемый интервал в 18 месяцев связан с прогнозом Давида Хауса из компании Intel, по мнению которого производительность процессоров должна удваиваться каждые 18 месяцев из-за сочетания роста количества транзисторов и быстродействия каждого из них

Четвертый способ

Повышение производительности ЭВМ связано с поиском форм представления чисел и алгоритмов, убыстряющих реализацию выполнения машинных операций. Однотактные операции убыстрению не поддаются. Предельные варианты модификации алгоритмов выполнения арифметических операций достигаются при конвейерных вычислениях, но это уже не ЭВМ, а параллельная ВС

Пятый способ требует заметной модернизации алгоритма управления вычислительными процессами и, следовательно, экономической и функциональной структуры ЭВМ. Самая главная инновация – конвейеризация вычислений

2) Принцип фиксированной структуры

Отсутствие возможности автоматического изменения структуры не позволяет в полной мере адаптировать ЭВМ к особенностям структуры решаемой задачи. Т. о. возможности по наращиванию производительности ЭВМ отсутствуют

3) Неоднородность ЭВМ

Этот принцип находится в резком противоречии с современной микро- и наноэлектроникой.

Вывод: Путь эволюционного совершенствования средств вычислительной техники на основе модели вычислителя и базовых принципах, перечисленных выше, не может привести к кардинальному улучшению их архитектурных характеристик.

Архитектурные особенности параллельных вычислительных систем

Вычислительная система – это средство обработки информации, функционирование которой основано на имитации работы не одиночных вычислителей, а коллектива вычислителей.

Четвертое поколение вычислительных средств (первое поколение ВС).

- Алгоритм управления вычислительными процессами в средствах четвертого поколения это универсальный параллельно-последовательный алгоритм с автоматическим изменением своей структуры. Структура вычислительных средств может также автоматически изменяться (программироваться) в зависимости от структуры и параметров решаемой задачи.

Пятое поколение вычислительных средств (второе поколение ВС).

- Это поколение вычислительных средств (Fifth-generation Computers) связано с решением еще более сложных (суперсложных) системных задач, известных под общим названием «проблемы искусственного интеллекта». Для решения задач такой сложности требуются самоорганизующиеся вычислительные средства, архитектура и функциональная структура которых должны допускать автоматические изменения универсального алгоритма управления процессом вычислении в течение всего времени решения задачи.

Архитектура вычислительных систем

5.1. Модель коллектива вычислителей

Логика развития средств обработки информации и дуализм понятия «вычислитель» порождают понятие «коллектив вычислителей». Допускается двойное толкование как коллектив людей-вычислителей и как коллектив аппаратурно-программных средств.

Принципы построения вычислительных систем

Коллектив аппаратурно-программных средств и называется вычислительной системой. Функционирование ВС основывается на структурной и функциональной имитации коллективов людей-вычислителей. Чем полнее такая имитация, тем богаче архитектурные возможности ВС. Каноническую основу ВС составляет пара

$$S =< H, A >$$

Где H – описание конструкции
A – алгоритм работы коллектива вычислителей
S – модель коллектива вычислителей

Это же – определение коллектива вычислителей

Конструкция дополняется следующим представлением:

$$H =< C, G >$$

$$C = \{c_i\}, i \in \{0, 1, \dots, N-1\}$$

C – множество вычислителей
G – описание структуры связей между вычислителями $c_i \in C$

Конструкция H основывается на основе следующих фундаментальных принципов:

1) Параллелизм при обработке информации, который требует параллельное функционирование вычислителей c_i, взаимодействующих через сеть G
2) Программируемая структура, т. е. настраиваемость структуры в зависимости от решаемой задачи. Программируемость определяет автоматическую настройку, т. е. по командам программ
3) Однородность
   Требует, чтобы коллектив вычислителей состоял из однородных и однотипных вычислителей, а также структура коллектива была тоже однородной

Принцип программируемости структуры требует выполнения следующего:

1) Возможность хранения изначальной физической структуры в памяти коллектива
2) Априорная настройка структуры, т. е. связи между вычислителями перед решением задачи
3) Возможность перенастройки структурных связей между вычислителями в процессе решения задачи

Структура ВС: типовые структуры сетей межвычислительных связей

Под структурой коллектива вычислителей понимается граф, вершинам которого сопоставлены вычислители c_i, а ребра – линии связи между ними. Проблема синтеза структур является нетривиальной.

структуры, которые нашли практическое применение (в том числе и простейшие):

1) Общая шина (0-мерная структура)

Использовались при формировании простейших ВС

Основной недостаток: ограниченность возможности по наращиванию количества вычислителей

2) Последовательное соединение (1-мерная структура)

Нет сосредоточенного ресурса, и каждый вычислитель имеет двух соседей

Ясно, что эта структура обладает большей надежностью и допускает одновременную реализацию между вычислителей

При построении промышленных систем использовалось кольцо

3) Связь «Решетка» (2-мерная структура)

Каждый вычислитель имеет четырех соседей. Такие структуры использовались при создании промышленных систем, как объединяющие процессоры, а также сейчас применяются при создании чипов, соединяющих множество процессоров. Как правило, внешние полюса отождествляются между собой по определенному правилу

В последние 2 десятилетия популярными стали структуры, в которых внешние полюса каждой строки и каждого столбца замыкаются. Такая структура называется «тор» (в геометрии под тором понимается тело, которое получается путем вращения круга вокруг прямой не пересекающей его и лежащей с ним в одной плоскости).

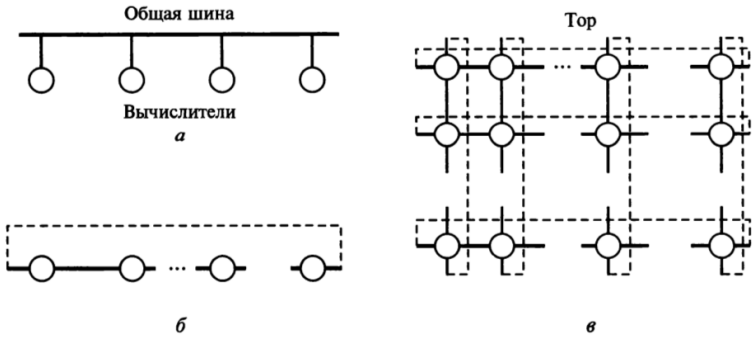


Рис. 3.1. Фрагменты простейших структур ВС:
а — нульмерная; *б* — одномерная; *в* — двумерная

В теории структур ВС тор – решетка, в которой имеет место отождествление граничных связей в каждой строке и в каждом столбце.

n-мерные структуры. В такой структуре каждый вычислитель имеет 2n соседей. Повышение размерности структуры приводит к повышению структурной надежности. При построении современных ВС широко используют гиперструктуры. Соответственно, гиперкуб – однородный граф, для которого справедливо $n = \log_2 N$, где N – количество вычислителей, n – количество связей для любого вычислителя. Соответственно, n – размерность гиперкуба.

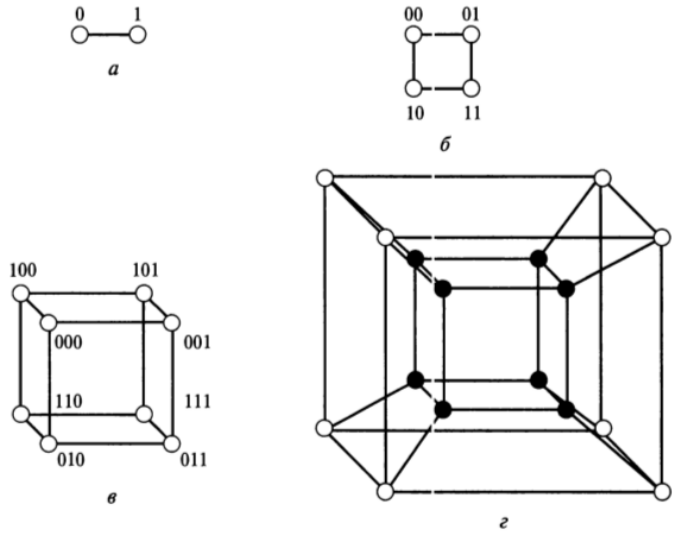


Рис. 3.2. Гиперкубические структуры ВС:
а — 1D-куб ($N = 2$, $n = 1$); *б* — 2D-куб ($N = 4$, $n = 2$); *в* — 3D-куб ($N = 8$, $n = 3$); *г* — 4D-куб ($N = 16$, $n = 4$)

Алгоритм функционирования ВС

Алгоритм A работы коллектива вычислителей С обеспечивает согласованную работу всех вычислителей $c_i \in C$

Алгоритм представляется A(P(D))

$$D = \bigcup_{i=0}^{N-1} D_i \text{ при } \bigcap_{i=0}^{N-1} D_i \neq \emptyset \ (*)$$

$D_i \in D$ - массив данных, распределенных на вычислитель c_i

P – параллельная программа $P = \bigcup_{i=0}^{N-1} P_i$, $\bigcap_{i=0}^{N-1} P_i = \emptyset$

P_i – ветвь параллельной программы

Условие (*) – условие информационной избыточности. Это условие позволяет создавать отказоустойчивые параллельные программы. Оно позволяет восстанавливать данные в случае отказа какого-либо вычислителя

Решение общей задачи происходит по некоторому параллельному алгоритму A, на основе которого пишется параллельная программа P. Распределенные по вычислителям данные D и параллельная программа определяют алгоритм функционирования A и, следовательно, работу конструкции H коллектива вычислителей.

$$S =< H, A >$$

Эквивалентное представление алгоритма работы коллектива вычислителей может быть в виде следующей суперпозиции:

$$(A_0 * A'_0) * \ldots * (A_i * A'_i) * \ldots * (A_{N-1} * A'_{N-1})$$

Где $(A_i * A'_i)$ определяет работу вычислителя c_i, т. е. как он должен функционировать среди других вычислителей. A_i определяет автономную работу вычислителя c_i, а A'_i определяет его взаимодействие с вычислителями $c_j = C/c_i$

$A'_i(P'_i(G))$

G – структура

P'_i - программа для установления связей и выполнения взаимодействий вычислителя c_i с вычислителями $c_j = C/c_i$

Разнообразие реализаций модели коллектива вычислителей определяется различием способов задания следующих элементов:

$G, \{P_i'\}, \{A_i'\}, 0 \le i < N-1$

$\{A_0', A_1', \dots, A_i', \dots, A_{N-1}'\}$, которые вместе со структурой G составляют среду для реализации межвычислительных взаимодействий, называют коммутаторами. Коммутатор столь же необходим при построении параллельных систем, как процессор и память для ЭВМ. При полном воплощении принципа однородности имеет место тождество

$$A_i \equiv A_j, A_i' \equiv A_j', i \ne j \in \{0, 1, \dots, N-1\}$$

Эти отношения эквивалентности обеспечивают высокую технологичность при проектировании и производстве параллельных ВС. Кроме того, они приводят к распределенному коммутатору, как объединению N локальных идентичных коммутаторов.

Модель вычислительной системы

$$S =< C, G, A\big(P(D)\big) >$$

Средства, основанные на модели коллектива вычислителей, называют ***вычислительной системой***. Наиболее архитектурно развитыми из них являются ВС с программируемой структурой. Модель коллектива вычислителей можно использовать на макро и микроуровнях. Представление коллектива вычислителей в качестве макровычислителя дает возможность создавать суперсложные или макросистемы. В свою очередь, вычислитель может быть представлен в виде коллектива микровычислителей.

C – множество вычислителей; G – структура сети связей между вычислителей; A – алгоритм работы множества C как коллектива вычислителей (взаимосвязанных через сеть G) при реализации параллельной программы P обработки данных D.

5.2. Техническая реализация модели коллектива вычислителей

Принципы технической реализации модели коллектива вычислителей

1. Модульность

Принцип, предопределяющий формирование ВС из унифицированных элементов, называемых модулями, которые функционально и конструктивно завершены и имеют средства сопряжения с другими элементами, и разнообразие которых составляет полный набор. При конструировании ВС с массовым параллелизмом достаточно пользоваться единственным модулем, обладающим функциональной и соединительной полнотой, в качестве которых используют элементарные процессоры (ЭП) и элементарные машины (ЭМ)

2. Близкодействие

Обуславливает такую организацию информационных взаимодействий между модулями-вычислителями, при которой каждый из них может обмениваться информацией с ограниченной частью других модулей-вычислителей.

3. Локальность связей. Это означает, что состояние $E_i(t+1)$ вычислителя c_i, I $\in$ {0,2,…,N-1}, на очередном временном шаге (t+1) зависит от состояний (на предшествующем шаге t) непосредственно с ним связанных вычислителей $c_i \in C^*$ т.е. состояние является функцией

$$E_i(t+1) = f(E_i(t), E_{i_1}(t), E_{i_2}(t), \dots, E_{i_M}(t))$$

Где i_l, $l = \overline{1,M}$ - номера вычислителей, составляющих C^*, M < N. При этом вычислители подмножества

$$C^* = \{c_{i_1}, c_{i_2}, \dots, c_{i_M}\}$$

Называются соседними по отношению к $c_i \in C$

4. Асинхронность функционирования ВС

Обеспечивается, если порядок срабатывания ее модулей определяется не с помощью вырабатываемых тем или иным образе м отметок времени, а достижением заданных значений определенны (кап с правило, логических) функций. Использование асинхронных схем позволяет достичь в системе алгоритмически предельного быстродействия: модули ВС срабатывают не-
медленно после выполнения соответствующего условия. Применение асинхронных схем обмена информацией между вычислителями позволяет не учитывать разброс в их тактовых частотах и колебания времени задержки сигналов в линиях связи.

5. Децентрализованность управления ВС

Достигается, если в системе нет выделенного модуля, который функционирует как единый для всей системы центр управления. децентрализованное управление системой основано на совместной работе всех исправных модулей системы, направленной на принятие решений, дос гавляющих оптимум выбранной целевой функции. Выполняя свою часть работы по выработке согласованного решения об управлении системой, каждый модуль пользуется только локальной информацией о системе.

6. Распределенность ресурсов ВС

Позволяет создавать такую систему, в которой нет единого ресурса, используемого другими в режиме разделения времени. Под ресурсами ВС понимаются все объекты, которые запрашиваются, используются и освобождаются в ходе выполнения вычислений. В качестве ресурсов ВС выступают процессоры или даже модули, входящие в их состав, модули оперативной памяти, внешние устройства, линии межмодульных связей, шины, файлы данных, компоненты ПО. Вместе с этим каждый ресурс распределенной ВС рассматривается как общий, доступный любому потребителю.

Архитектурные свойства ВС

- Масштабируемость ВС.

Под масштабируемостью ВС понимается их способность к наращиванию и сокращению ресурсов, возможность варьирования производительности

- Универсальность ВС.

Вычислительные системы алгоритмически и структурно универсальны. Принято считать, что ЭВМ (основанные на модели вычислителя) являются алгоритмически универсальными, если они обладают способностью (без изменения своих структур) реализовать алгоритм решения любой задачи. Структурная универсальность ВС является следствием воплощения архитектурных принципов коллектива вычислителей, в частности принципа программируемости структуры. Суть этого принципа возможность автоматически (программно) порождать специализированные (проблемно-ориентированные) виртуальные конфигурации, которые адекватны структурам и параметрам решаемых задач.

- Производительность ВС.

В отличие от ЭВМ, построенных на основе модели вычислителя, ВС не имеют принципиальных ограничений в повышении производительности. Увеличение производительности в них достигается за счет не только повышения физического быстродействия микроэлектронных элементов, а главным образом увеличения числа вычислителей.

- Реконфнгурнруемоеть ВС

Статическая реконфигурация ВС обеспечивается: варьированием числа вычислителей, их структуры и состава; выбором дня вычислителей числа полюсов для связи с другими вычислителями; возможностью построения структур в виде графов, относящихся к различным классам; допустимостью применения в качестве связен каналов различных типов, различной физической природы и различной протяженности и т. п. Благодаря приспособленности ВС к статической реконфигурации достигается адаптация системы под область применения на этапе ее формирования.

Динамическая реконфигурация ВС поддерживается возможностью образования в системах таких (виртуальных) подсистем, структуры и функциональные организации которых адекватны входной мультипрограммной ситуации и структурам решаемых задач.

- Надежность и живучесть ВС

Под надежностью ВС понимается ее способность к автоматической(программной) настройке и организации функционирования таких структурных схем, которые при отказах и восстановлении вычислителей обеспечивают заданный уровень производительности или, говоря иначе, возможность использовать фиксированное число исправных вычислителей

Под живучестью ВС понимают свойство программной настройки и организации функционирования таких структурных схем, которые в условиях отказов и восстановления вычислителей гарантируют при выполнении параллельной программы производительность в заданных пределах или возможность использования всех исправных вычислителей.

- Самоконтроль и самодиагностика ВС.

Заключение об исправности или неисправности отдельных вычислителей системы принимается коллективно всеми вычислителями на основе сопоставления их индивидуальных заключений об исправности соседних с ними вычислителей.

Технико-экономическая эффективность ВС.

Конструктивная однородность позволяет резко сократить сроки разработки и изготовления систем, обеспечивает высокую технологичность производства, упрощает и статическую, и динамическую реконфигурации ВС, облегчает их техническую эксплуатацию.

5.3. Параллельные алгоритмы

Элементарные понятия параллельного программирования

Описание процесса обработки информации, ориентированного на реализацию в коллективе вычислителей, называют *параллельным алгоритмом* (ПА). ПА, в отличие от последовательного, предусматривает реализацию на каждом временном шаге множество операций и, как и последовательный, сохраняет зависимость последующих этапов вычислений от результатов предыдущих. Запись алгоритма на языке, доступном коллективу вычислителей, называют *параллельной программой*. А этот язык называют

языком параллельного программирования. Методы вычислительной математики и соответствующего алгоритма, как правило, последовательный процесс приспособления последовательных алгоритмов к решению в коллективе вычислителей, называется *распараллеливанием*. Теоретическая и практическая деятельность, направленная на создание параллельных алгоритмов и программ, называется *параллельным программированием*. Качество параллельных алгоритмов и программ определяется *методикой распараллеливания*. Существует два подхода:

1) Локальное

Алгоритм решения задачи расщепляется на предельно простые блоки: операторы или операции, и требуется максимальное выделение этих операций для каждого шага вычислений. Этот подход обеспечивает неоднородную загрузку вычислителей, поэтому считается неэффективным. Его следует применять при оценке предельных возможностей по распараллеливанию. Практический эффективный подход – крупноблочное распараллеливание.

2) Глобальное (блочное)

Требует расщепление вычислительного процесса на крупные блоки (подзадачи), между которыми существует слабая связь. Последнее обеспечивает относительно незначительные расходы времени на реализацию обменов. Такие подзадачи называют *ветвями параллельной программы*

$$\boldsymbol{P} = \bigcup_{\boldsymbol{i=1}}^{\boldsymbol{n}} \boldsymbol{P_i}, n - \text{количество ветвей}$$

Основной прием – распараллеливание по циклу. В пределе такое распараллеливание дает количество ветвей, равное числу выполнений цикла. Заметим, что в параллельной программе всегда требуется выполнять обмены информацией между ветвями. В параллельных программах также присутствуют последовательные участки (например, в начале и конце при вводе и выводе информации). В параллельной программе последовательные участки могут быть также внутри – они соответствуют негрупповым обменам информацией.

Параллельный алгоритм умножения матриц

Анализ прямых и итерационных методов в вычислительной математике показывает, что в них, как правило, используются операции умножения векторов и матриц данных. Разработаем алгоритм для умножения матриц большого размера.

$$A[1..N; 1; k] * B[1..k; 1..M] = C[1..N; 1..M]$$

Пусть эта операция реализуется на n вычислителях.

$$\begin{matrix} a_{11} & \dots & a_{1h} & \dots & a_{1k} \\ a_{i1} & \dots & a_{ih} & \dots & a_{ik} \\ a_{N1} & \dots & a_{Nh} & \dots & a_{NN} \end{matrix} * \begin{matrix} b_{11} & \dots & b_{1j} & \dots & b_{1M} \\ b_{i1} & \dots & b_{hj} & \dots & b_{hM} \\ b_{k1} & \dots & b_{kj} & \dots & b_{MM} \end{matrix} = \begin{matrix} c_{11} & \dots & c_{1j} & \dots & c_{1M} \\ c_{i1} & \dots & c_{ij} & \dots & c_{iM} \\ c_{N1} & \dots & c_{Nj} & \dots & c_{MM} \end{matrix}$$

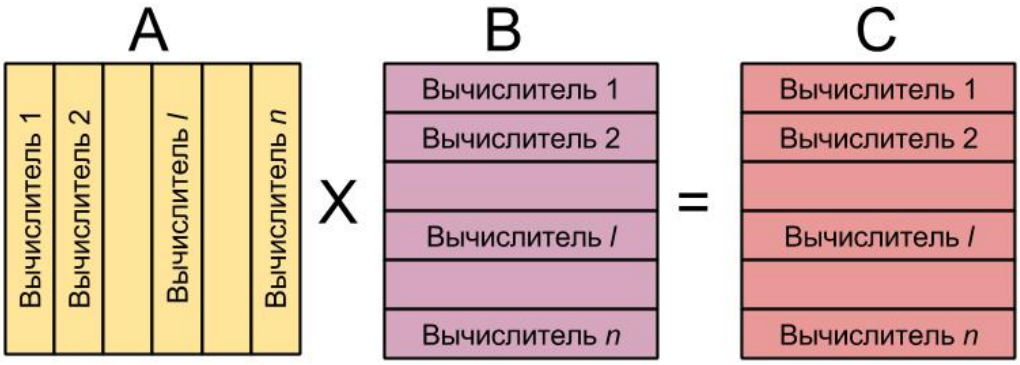
Пусть $N * K \gg n, K * M \gg n$

Заметим, что приведенные выше условия являются необходимыми.

$$C_{ij} = \sum_{h=1}^{K} a_{ih} * b_{hj}, \qquad 1 \le i \le N; 1 \le j \le M$$

Вообще, распараллеливание следует применять для тех задач, решение для которых недоступно на ЭВМ. Такие задачи называют **сложными** или **трудоемкими**. При параллельной обработке необходимо, чтобы каждый вычислитель рассчитывал n-ую часть элементов матрицы C. При этом следует заметить, что размещение матриц A и B в каждом вычислителе требует большой суммарной емкости памяти. Необходимо каждому вычислителю выделить 1/n часть матриц A и B, т. е. необходимо произвести однородное распределение этих матриц по вычислителям. Например, этого можно достичь, если матрицу A разделить на n строк, B разделить на n столбцов, и каждая такая строка и столбец будут расположены в своем вычислителе, т. е. в l-ом вычислителе размещаются следующие строки матрицы A

$$A: (l-1)]\frac{N}{n}[+1, (l-1)]\frac{N}{n}[+2, \dots, l]\frac{N}{n}[$$
$$B: (l-1)]\frac{M}{m}[+1, (l-1)]\frac{M}{m}[+2, \dots, l]\frac{M}{m}[$$



Где $]x[$ - ближайшее к x такое целое число, что выполняется неравенство $]x[\geq x$

Параллельный вычислительный процесс можно организовать следующим образом: первый вычислитель высылает свою первую строку из полосы матрицы A всем остальным вычислителям. После этого все вычислители параллельно рассчитывают свои элементы первой строки матрицы C. Затем первый вычислитель рассылает остальным вычислителям вторую строку из своей полосы матрицы A. Затем остальные вычислители параллельно рассчитывают свои элементы второй строки матрицы C. Рассылка элементов из первого вычислителя заканчивается после передачи $]\frac{N}{n}[$. Затем рассылкой начинает заниматься второй вычислитель и т. д. Процесс завершается после рассылки и обработки последней строки из полосы матрицы A n-го вычислителя.

Применяя такое однородное распределение исходных данных и процедуры обработки, получаем параллельный алгоритм, состоящий из идентичных ветвей. Построим блок-схему для этой ветви параллельной программы. При этом пусть α – номер передающей машины. Тогда 1, 2, …, α-1, α+1, …, n – номера вычислителей приемников.

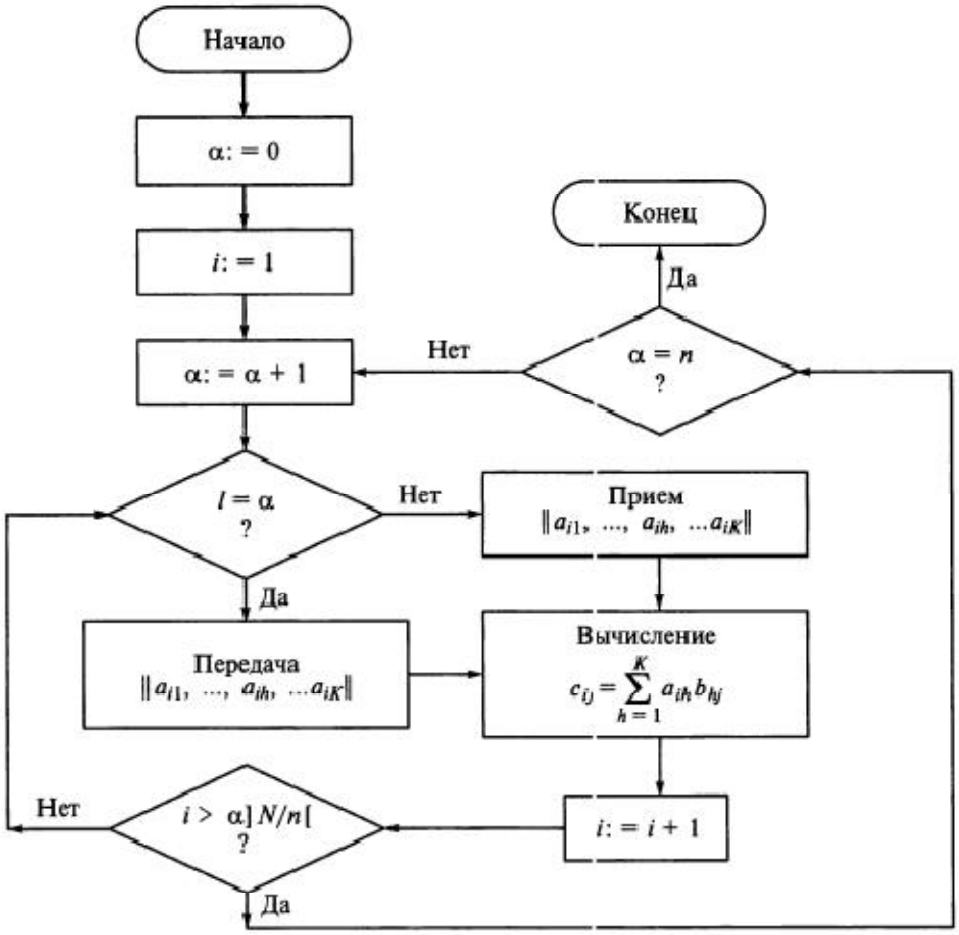


Рис. 3.5. Схема ветви параллельного алгоритма умножения матриц:

α — номер передающего вычислителя; {1, 2, …, α − 1, α + 1, …, n} — номера принимающих вычислителей; $]M/n[(l-1) < j \leqslant]M/n[l$

Показатели эффективности параллельных алгоритмов: коэффициенты накладных расходов, ускорение и эффективности

Каждая ветвь параллельной программы есть последовательность обычных операторов: арифметических и логических, а также операторов обмена данными. Ясно, что чем меньше времени будут расходы на обмены информацией, тем выше эффективность параллельной программы. Следовательно, в качестве одного из показателей эффективности можно использовать коэффициент накладных расходов

$$\varepsilon = \frac{t}{T}$$

Где t – время, расходуемое ВС на организацию и реализацию; T – время, расходуемое на вычисления

Оценим этот показатель для алгоритма умножения матриц

- Передается K слов
- Всего умножений будет $K]\frac{M}{n}[$; сложений - $(K-1)]\frac{M}{n}[$
- Т. к. K>>n, то можно считать, что на каждое переданное слово $\rho =]\frac{M}{n}[$ операций умножения и сложения
- Т. о. коэффициент накладных расходов $\varepsilon = \frac{t_{\text{п}}}{\rho(t_{\text{с}}+t_{\text{у}})}$

- Где $t_{\text{п}}$ – время пересылки, $t_{\text{с}}, t_{\text{у}}$ – время соответственно сложения и умножения
- Следовательно, минимальное значение $\rho = 1$

$$\varepsilon_{max} = \frac{t_{\text{п}}}{(t_{\text{с}} + t_{\text{у}})}$$

Очевидно, $\rho \to \infty$ (при увеличении размеров матрицы $M \to \infty$) $\varepsilon \to 0$, т. е. накладные расходы становятся пренебрежимо малыми

Данный факт раскрывает суть методики крупноблочного распараллеливания

Чтобы получить эффективный параллельный алгоритм, необходимо, чтобы в каждом вычислителе обрабатывалось как можно больше данных

Коэффициент ускорения

$$\chi = \frac{\tau_1}{\tau_n}$$

Здесь τ_1 – время решения задачи на одном вычислителе при использовании последовательной программы; τ_n – время выполнения соответствующего Р-алгоритма в системе из *n* вычислителей

Коэффициент ускорения χ говорит о том, насколько быстрее коллектив машин решает задачу, чем одна машина

Коэффициент эффективности

$$E = \frac{\chi}{n}$$

Имеет место неравенство $\chi \leq n, E \leq 1$

Это объясняется тем, что происходят расходы времени на обмен информацией

Одной из целей распараллеливания сложных задач является достижение максимума коэффициента ускорения:

$$\max \chi = n$$
$$\max E = 1$$

Парадокс параллелизма

«Парадокс» параллелизма заключается в нелинейном росте производительности от увеличения количества вычислителей

$$\chi > n, E > 1 \ (*)$$

Эффект был обнаружен в Отделе вычислительных систем института математики СО АН СССР в 60-70-ых годах при решении широкого круга вычислительных стохастических и информационно-логических задач на советских ВС с программируемой структурой. Этот эффект постоянно шокирует специалистов, но по сути парадоксом не является. Этот эффект объясняется следующим:

1) Эффект памяти

Современная ВС – коллектив вычислителей, каждый из которых имеет свою локальную память. Сеть связей между локальными вычислителями делает их память общедоступной. Если суммарная локальная память имеет достаточно большую емкость и допускает вложение параллельного алгоритма и данных и, следовательно, не приходится использовать более медленную внешнюю память, то будет достигнут минимум времени τ_n. С другой стороны, если сложность задачи достаточно высока и ее невозможно разместить в локальной памяти одного вычислителя, и если она допускает решение на

отдельном вычислителе только при использовании внешней памяти, то будет достигнуто значение времени τ_1, отличающееся от минимального. Если время передачи между вычислителями сравнимо с временем обращения к локальной памяти вычислителя, то будет иметь место следующее равенство:

$$\tau_1 = A_n * n * \tau_n, A_n > 1$$

Следовательно, будет справедливо неравенство (*)

2) Возможность применения априори параллельных методов решения задач, решение которых на последовательных ЭВМ невозможно

Понятие о сложных задачах

Эффективность реализации параллельной программы на ВС зависит от объема операций, которые следует выполнить.

$$\varepsilon(V, n) = \frac{t(V, n)}{T(V, n)}$$

V – количество операций, которое следует выполнить при решении задачи
n – количество вычислителей

Установлено, что при n = const $\varepsilon(V, n) \to 0$ при $V \to \infty$
Значение $\varepsilon(V, n)$ будет практически удовлетворительным, если выполняется следующее равенство:

$$V = n * 10^k, k \geq 1 \ (**)$$

k - эмпирический коэффициент; зависит от быстродействия каналов связи между вычислителями

Если это быстродействие равно скорости обращения к локальной оперативной памяти, то можно взять k=1

Т. о. если объем операций превышает хотя бы на порядок количество вычислителей n, то такую задачу лучше всего решать на системе. Задачу, удовлетворяющую (**) , принято считать ***сложной (трудоемкой, с большим объемом вычислений)***. Задача считается простой, если она может быть решена на одном вычислителе.

Схемы обмена информаций между ветвями параллельных алгоритмов

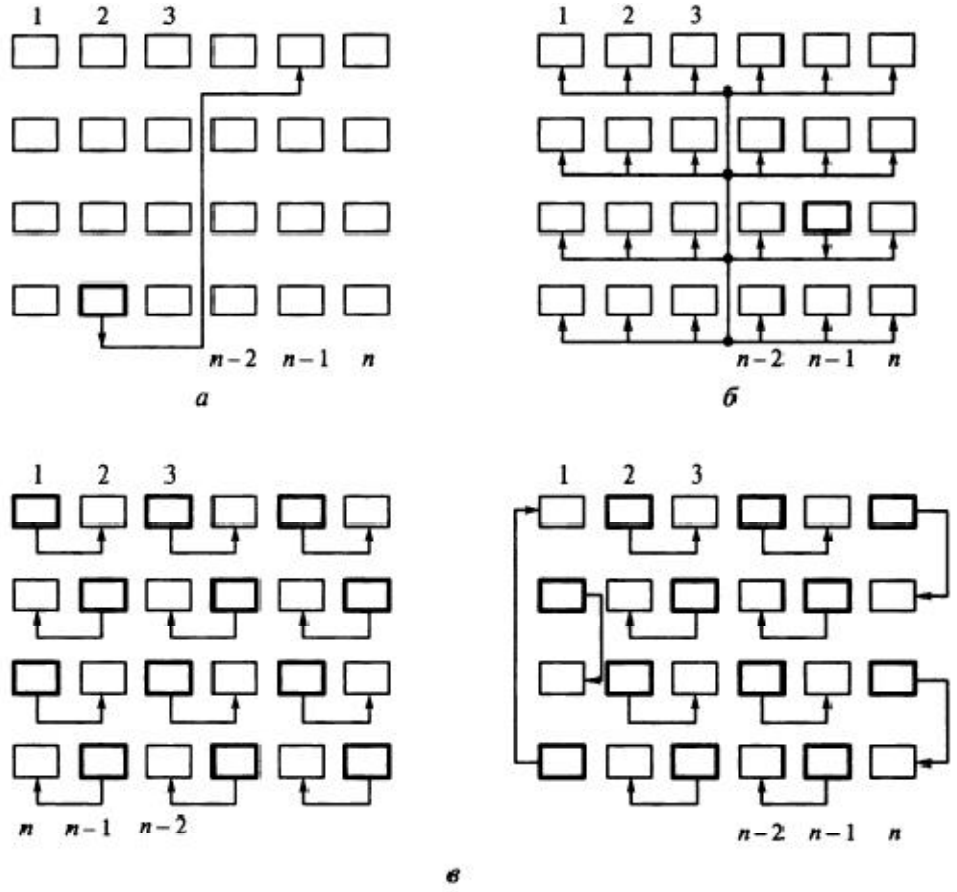


Рис. 3.6. Схемы обмена информацией между ветвями параллельного алгоритма:
а — дифференцируемый обмен; *б* — трансляционный обмен; *в* — конвейерно-параллельный обмен; □ — приемник; □ — передатчик

Параллельный алгоритм – композиция ветвей, связанных для осуществления обменов. Следовательно, возникает вопрос о разнообразии схем обмена информацией. Ясно, что любая схема может быть реализована при помощи дифференцированного обмена.

1) Дифференцированный обмен

Информация из одной ветви передается в другую. Соответственно, каждая ветвь реализуется на своем вычислителе. Ясно, что при таком обмене участвуют только две ветви, остальные простаивают. Такой обмен неэффективен.

Возникает вопрос: существуют ли в параллельных алгоритмах групповые или коллективные обмены, при которых все вычислители работают одновременно? Анализ параллельных алгоритмов решения сложных задач показывает, что к таковым обменам относятся:

2) трансляционный обмен (One-to-All Broadcast)

из одного вычислителя информация передается всем остальным. Заметим, что данная схема впервые выведена в 60-х годах в институте математики СО АН СССР и позднее обнаружена на западе;

3) трансляционно-циклический обмен (All-to-All Broadcast)

суть обмена заключается в повторении n раз трансляционного обмена из каждого вычислителя;

4) конвейерно-параллельный обмен

реализуется в 2 этапа:

1) например, при четном n на первом этапе информация из нечетных вычислителей передается в соседние вычислители с четными номерами;
2) из четных вычислителей информация передается в вычислители с нечетными номерами с большим номером.

Встречается также коллекторный обмен. По сути, он является инвертированным трансляционным обменом, при котором информация собирается в одном вычислителе из всех остальных. Такой обмен сложно реализовать технически. Поэтому его реализуют с помощью (n-1) дифференцированных обменов. Следовательно, при коллекторном обмене в любой момент времени работают только 2 вычислителя, остальные простаивают.

Статистика обменов

Тип обмена	ДО	ТО	ТЦО	КПО	КО
Частота использования	2%	17%	40%	34%	7%

Опыт применения методики крупноблочного распараллеливания сложных задач

Опыт применения отечественных ВС с программируемой структурой позволяет сделать следующие выводы:

1) Сложные задачи допускают представление в виде параллельных алгоритмов с идентичными ветвями. Следовательно, проблемы программирования сводятся к написанию лишь одной ветви
2) При распределении исходных данных между ветвями параллельного алгоритма эффективным является принцип однородного распределения массива
3) Для построения параллельных алгоритмов достаточно использовать 5 схем обмена, рассмотренных ранее
4) Простота схем обмена между ветвями позволяет ограничиться только однородными структурами, а последнее дает высокую технико-экономическую эффективность ВС
5) Для записи параллельных алгоритмов решения сложных задач эффективной версией языков программирования FORTRAN, C и других такие версии языков называют параллельными. Параллельные языки от последовательных отличаются тем, что в них включены операторы для организации взаимодействий между ветвями. Например, могут быть введены операторы для реализации схем обмена. Дополнение к транслятору для параллельных версий языков оценивается в несколько процентов
6) Простота схем обмена и распределения данных по ветвям ведет к простоте написания параллельных программ. При этом трудозатраты на написание параллельных программ, в отличие от последовательных, не превышает 10%
7) Схемы ТО, ТЦО и КПО составляют более 90% от всех схем. Параллельные алгоритмы, построенные на методике крупноблочного распараллеливания, характеризуются несколькими процентами затрат времени на обмен по сравнению с общим временем решения задач.

5.4. Концептуальное понятие о вычислительных системах

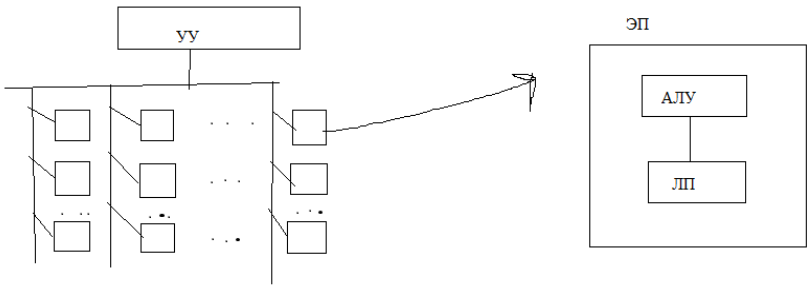
Понятие о вычислительных системах

Дать достаточно четкое понятие системы нельзя. В энциклопедическое понятие система переводится с греческого как «целое состоит из частей», т. е. система – множество элементов, находящихся в отношениях и связях друг с другом, которые образуют определенную целостность и единство. Под системой понимается средство обработки информации, основанные на модели коллектива вычислителей. При достаточно общей трактовке под вычислительной системой понимается совокупность взаимосвязанных и одновременно функционирующих аппаратурно-программных вычислителей, которые способны не только реализовать параллельный процесс решения сложных задач, но и априори в процессе решения автоматически настраиваться и перенастраиваться с целью достижения адекватности между своей функционально-структурной организацией и структурой и параметрами решаемой задачи

Типы архитектур: MISD, SIMD, MIMD

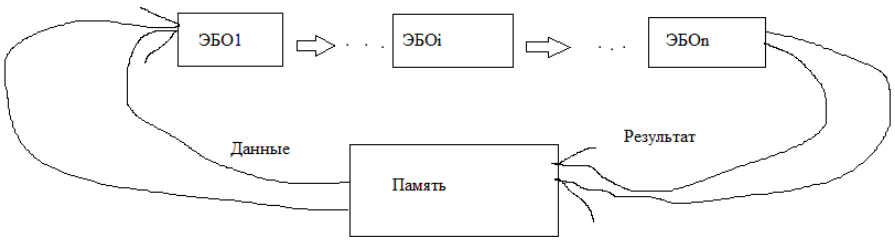
Флин предложил классификацию по потокам команд и данных, имеющих место в вычислительных средствах. Под потоком команд понимается любая их последовательность, подлежащая выполнению на вычислительном средстве. Для исполнения команд требуются данные, следовательно, поток команд порождает поток данных.

1) SIMD



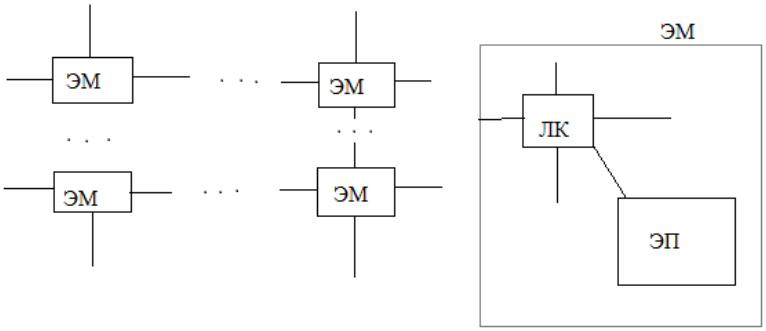
Единое устройство управления, где хранится программа вычислений, множество одинаковых элементарных процессоров. Локальная память предназначена для хранения порций данных. УУ направляет поток команд, которые выполняются одновременно всеми ЭП, но каждый над своими данными.

2) MISD



Такие ВС – предел модификации последовательной ЭВМ. В ней данные пропускаются через цепочку элементарных блоков обработки (конвейер). Каждый ЭБО реализует свою операцию, и все они работают параллельно. За счет этого достигается рост производительности. Практически все современные процессоры используют конвейер

3) MIMD



Множество элементарных машин, связанных между собой через сеть межмашинных линий. Каждая ЭМ – композиция локального коммутатора и элементарного процессора. Локальный коммутатор обеспечивает связь с соседними ЛК. ЭП имеет полноценное УУ. Его локальная память предназначена для хранения не только части данных, но и ветви параллельной программы. В архитектуре этого класса имеются следующие виды систем:

мультипроцессорные, распределенные, с программируемой структурой, кластерные.

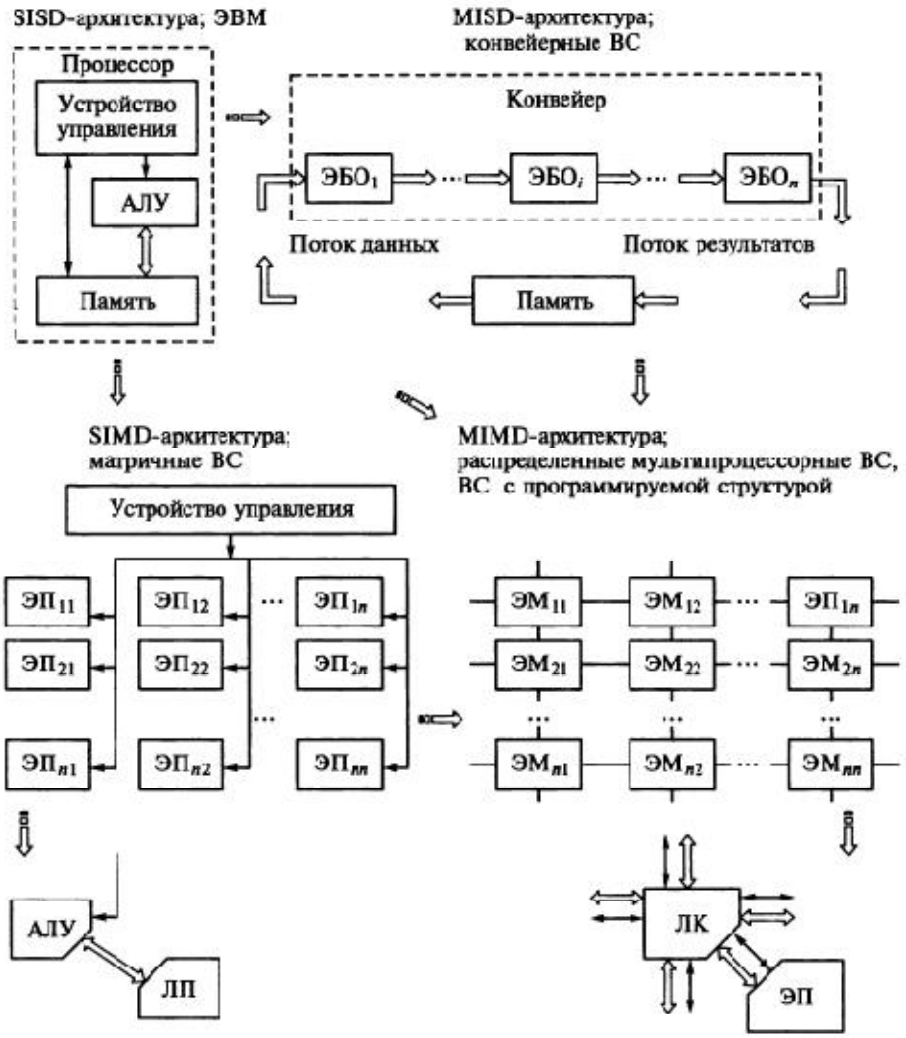


Рис. 3.7. Развитие архитектуры вычислительных средств:

АЛУ — арифметико-логическое устройство; ЭБО — элементарный блок обработки; ЭП — элементарный процессор; ЭМ — элементарная машина; ЛП — локальная память; ЛК — локальный коммутатор; → — поток команд; ⇨ — поток данных; ⇢ — направление трансформации архитектуры

Классификация ВС

Мультипроцессорные вычислительные системы

МВС представляют собой множество процессоров и некий единый ресурс (как правило, общая память). Взаимодействие процессоров с памятью осуществляется через коммуникационную среду, а между процессорами – через общую память.

Распределенные вычислительные системы

В таких системах все ресурсы представляются как композиция неких функционально завершенных элементарных машин.

ВС с программируемой структурой

Самый архитектурно совершенный класс систем. Обладает возможностью автоматически подстраиваться под структуры и классы решаемых задач, т. е. в таких системах могут быть порождены виртуальные специализированные ВС. Концепция таких систем реализована в 70-х годах.

Кластерные ВС

Понятие введено в компании DEC. По их определению, кластер – группа компьютеров, связанных между собой и функционирующих как единое средство обработки информации. Могут иметь любую архитектуру, кроме SISD. Главным образом, они строятся из серийных аппаратурно-программных средств, но могут использовать и специально разработанные компоненты.

Транспьютерная ВС

Композиция из одинаковых взаимосвязанных микропроцессоров, называемых транспьютерами. В состав транспьютера входят локальные процессор и память, а также локальные средства коммутации и линки (Link связь), позволяющие организовать взаимодействия с другими транспьютерами.

Матричные ВС

Основываются на принципе массового параллелизма, в них обеспечивается возможность одновременной реализации большого числа операций на элементарных процессорах (ЭП), объединенных в матрицу.

Конвейерные ВС

Системы, архитектура которых является предельным вариантом эволюционного развития последовательной ЭВМ и простейшей версией модели коллектива вычисли"елей. В основе таких систем лежит конвейерный (или цепочечный) способ обработки информации, а их функциональная структура представляется в виде последовательности связанных элементарных блоков обработки (ЭБО) информации. Все блоки работают параллельно, но каждый из них реализует лишь свою операцию над данными одного и того же потока.

Конвейерные вычислительные системы

6.1. Каноническая функциональная структура конвейерного процессора

Назначение конвейерного процессора и Векторные операции

В конвейерных ВС основной объем операций по обработке данных выполняется одним или несколькими конвейерными процессорами. Конвейер оперирует с векторами данных, которые являются одномерными массивами или, в терминах алгебры, строкой или столбцом.

$$A = (A_1, A_2, \dots, A_n)$$

В конвейере векторные операции реализуются аппаратурно. Здесь рассматриваются операции покомпонентного умножения, деления, сложения и вычитания, либо формирование вектора из чисел, обратным компонентам данного вектора. Для более сложных операций могут быть введены свои векторные команды.

$$A + \alpha B, (A + \alpha)B,$$

где A и B – векторы, α – скаляр (число, константа).

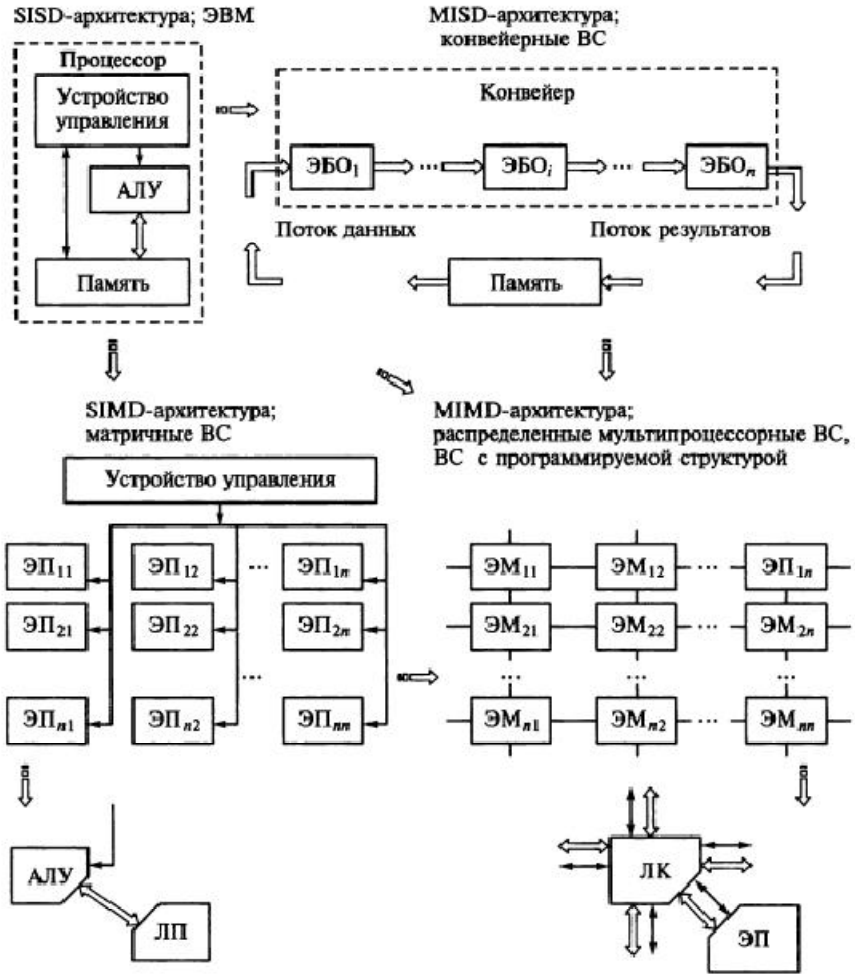


Рис. 3.7. Развитие архитектуры вычислительных средств:

АЛУ — арифметико-логическое устройство; ЭБО — элементарный блок обработки; ЭП — элементарный процессор; ЭМ — элементарная машина; ЛП — локальная память; ЛК — локальный коммутатор; → — поток команд; ⇨ — поток данных; ▫⇨ — направление трансформации архитектуры

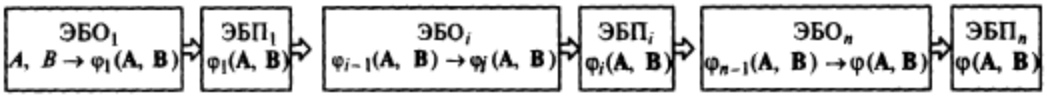
MISD-архитектура

Такие ВС – предел модификации последовательной ЭВМ. В ней данные пропускаются через цепочку элементарных блоков обработки (конвейер). Каждый ЭБО

реализует свою операцию, и все они работают параллельно. За счет этого достигается рост производительности. Практически все современные процессоры используют конвейер

Структура и функционирование конвейерного процессора

Каноническая структура



Функционирование

$A_1, B_1 \to \varphi_1(\mathbf{A}_1, \mathbf{B}_1)$				
$A_2, B_2 \to \varphi_1(\mathbf{A}_2, \mathbf{B}_2)$	⋯			
	⋯			
$A_i, B_i \to \varphi_1(\mathbf{A}_i, \mathbf{B}_i)$	⋯	$\varphi_{i-1}(\mathbf{A}_1, \mathbf{B}_1) \to \varphi_i(\mathbf{A}_1, \mathbf{B}_1)$		
$A_{i+1}, B_{i+1} \to \varphi_1(\mathbf{A}_{i+1}, \mathbf{B}_{i+1})$	⋯	$\varphi_{i-1}(\mathbf{A}_2, \mathbf{B}_2) \to \varphi_i(\mathbf{A}_2, \mathbf{B}_2)$	⋯	
	⋯		⋯	
$A_n, B_n \to \varphi_1(\mathbf{A}_n, \mathbf{B}_n)$	⋯	$\varphi_{i-1}(\mathbf{A}_{n-i+1}, \mathbf{B}_{n-i+1}) \to$ $\to \varphi_i(\mathbf{A}_{n-i+1}, \mathbf{B}_{n-i+1})$	⋯	$\varphi_{n-1}(\mathbf{A}_1, \mathbf{B}_1) \to \varphi(\mathbf{A}_1, \mathbf{B}_1)$
$A_{n+1}, B_{n+1} \to \varphi_1(A_{n+1}, B_{n+1})$	⋯	$\varphi_{i-1}(A_{n-i+2}, B_{n-i+2}) \to$ $\to \varphi_i(A_{n-i+2}, B_{n-i+2})$	⋯	$\varphi_{n-1}(\mathbf{A}_2, \mathbf{B}_2) \to \varphi(\mathbf{A}_2, \mathbf{B}_2)$
	⋯		⋯	

Рис. 4.1. Конвейерный процессор:
ЭБО — элементарный блок обработки информации; ЭБП — элементарный блок памяти;
A, B — векторы-операнды; $\varphi_i(\mathbf{A}, \mathbf{B})$ — частичное преобразование векторов **A, B**

Конвейер в общем случае реализуется как цепочки из элементарных компонентов обработки информации и памяти. Каждый из блоков ЭБО осуществляет частичное преобразование векторов-операндов.

$$A, B \to \varphi_1(A, B) \to \cdots \to \varphi_i(A, B)$$

Результат преобразований операндов сохраняется в элементарных блоках памяти ($ЭБП_i$). Блоки $ЭБП_i, i = \{1, n-1\}$, и блок $ЭБП_n$ используются для хранения промежуточных результатов и могут быть объединены в единое целое, либо оперативную память, либо векторные регистры. В простейшем случае элементарные блоки обработки конвейера могут реализовывать отдельные фазы операций, т. е. выполнять микрооперации. Например, при сложении двух вещественных чисел, представленных с плавающей запятой, выполняются следующие микрооперации:

1) Сравнение порядков
2) Выравнивание порядков
3) Сложение мантисс
4) Нормализация

Элементы векторов подаются в конвейер в дискретные моменты времени и в соответствии с их расположением в векторах. На каждом шаге в $ЭБО_1$ заносится новая пара элементов-операндов A и B. В $ЭБО_i, i = \{2, n\}$ заносится информация из $ЭБО_{i-1}$. Процесс вычисления $\varphi(A, B)$ для пары элементов A и B разделен на n этапов. Все блоки конвейера работают параллельно, но каждый из них реализует свой этап операций и обрабатывает свои элементы в фиксированный момент времени t. Очевидно, что время обработки на конвейере конкретных элементов векторов равно суммарному времени на преобразование во всех ЭБО.

6.2. Конвейерные системы типа «типа память-память»

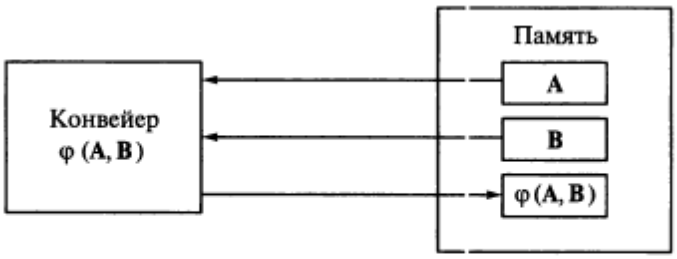
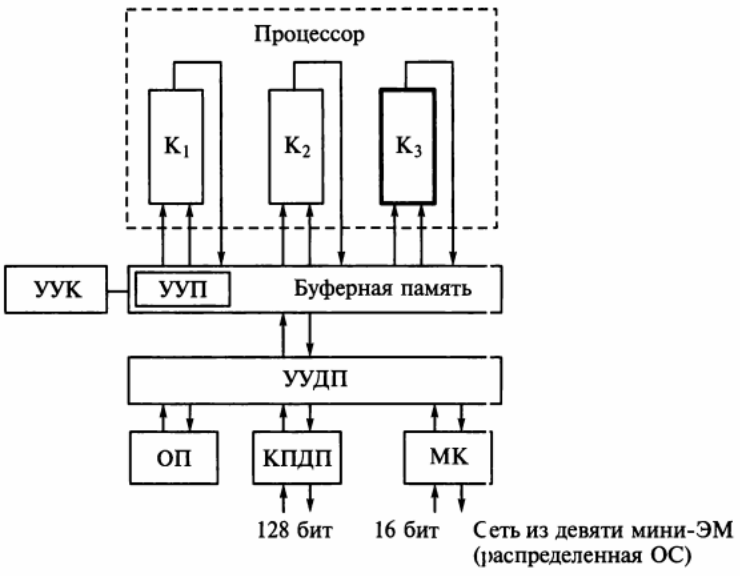


Рис. 4.2. Архитектура ВС типа «память-память»

Начиная с 1973 года, фирма CDC выпустила ряд конвейерных ВС с архитектурой типа «память-память». В таких ВС элементы-операнды векторов A и B, необходимые для выполнения векторных команд выбираются непосредственно из оперативной памяти и результат записывается в ту же память. В качестве преобразования $\varphi(A,B)$ может быть результат одной из арифметических операций над элементами векторов A и B. Для хранения векторов A и B и результата $\varphi(A,B)$ требуются области памяти одинаковой емкости. Примеры ВС: STAR-100, CYBER-203 и CYBER-205.

Система STAR-100(String Array computer) фирмы CDC(Control Data Corporation)

Разрабатывалась CDC в 1965-73 гг. Быстродействие – 100 млн опер/с. Стоимость - порядка 15 млн долларов. Состояла из двух подсистем: первая осуществляла обработку данных, вторая обеспечивала функционирование операционной системы. Ядро первой подсистемы – процессор, образуемых из нескольких конвейеров. В типовых конфигурациях системы было 3 конвейера K_1, K_2, K_3. Конвейеры были специализированы. K_1, K_2 служили для выполнения векторных операций. K_3 использовался для реализации операций над скалярными операндами, т. е. K_1, K_2 – конвейеры, каждый из которых служил для выполнения операций с плавающей запятой над парами векторов данных. K_3 – конвейер для обработки обычных операндов в неорганизованных векторах. K_1, K_2 определили уровень быстродействия системы. Конвейеры имели программируемую структуру, следовательно, в них могли выполняться различные арифметические операции, но для начала новой операции требовалась перенастройка.



Функциональная структура STAR-100

УУК – устройство управления командами
УУП – устройство управления потоками
УУДП – устройство управления доступом к памяти
ОП – оперативная память
КПДП – канал прямого доступа в память
МК – мультиплексный канал

Каждый конвейер мог включать в себя до 30 блоков. Все блоки работают параллельно. Любой конвейер воспринимал 64-хразрядный код либо как один 64-хразрядный операнд, либо 2 32-хразрядных. Время выполнения операций над одной парой операндов в любом из блоков конвейеров не превышала 40 нс. Следовательно, данные могли поступать в процессор со скоростью 100 млн операций в секунду. STAR-100 имела набор из 230 команд, из которых 65 - для работы с векторами данных и 130 – для работы со скалярами. Средство управления подсистемой обработки данных представлено композицией из УУК, УУП и УУДП.

УУК имело буфер опережающего просмотра команд емкостью в 4 512-разрядных суперслова со стековым механизмом работы.

УУП использовалось для управления потоками операндов и команд между УУДП, конвейерами и УУК.

В ОП хранились программа и данные. Реализована на магнитных сердечниках и имела емкость 8 Мбайт. Имелись четыре виртуальных канала обращения к памяти, которые
реализовывались устройством управления доступом к памяти. Два канала использовались для чтения операндов для конвейеров K_1, K_2, один канал для записи результатов, и один канал – для устройства ввода-вывода.

Буферная память была введена вследствие того, что быстродействие оперативной памяти было существенно ниже быстродействия процессора. Она представляет собой совокупность регистров с временем цикла 40 нс.

Операционная система относилась к классу распределенных. Ее функции, включая управление внешними запоминающими устройствами и устройствами ввода-вывода информации, реализовывались специальной вычислительной сетью из девяти мини-машин. Система программирования STAR-100 включала компиляторы с языков APL-STAR, COBOL и FORTRAN.

Семейство систем Cyber

Модернизированный вариант STAR-100. Производительность CYBER-203 – 100 млн опер./с. Являлась конвейерной, имела ту же самую систему команд, полностью совместимое ПО с STAR-100. Содержала обычный скалярный процессор за место K_3, что обеспечивало 6х увеличение быстродействия при скалярной обработке информации. Емкость оперативной памяти – 16 Мбайт. Скорость выборки из памяти – 100 млрд бод (разрядность слов – 64).

CYBER-205 обладала более современной архитектурой в сравнении с 203. В системе допускалось варьирование количества конвейеров от одного до четырех. Пиковая производительность – 200 млн опер./с., ОП – 32 Мбайт. Все конвейеры могли работать

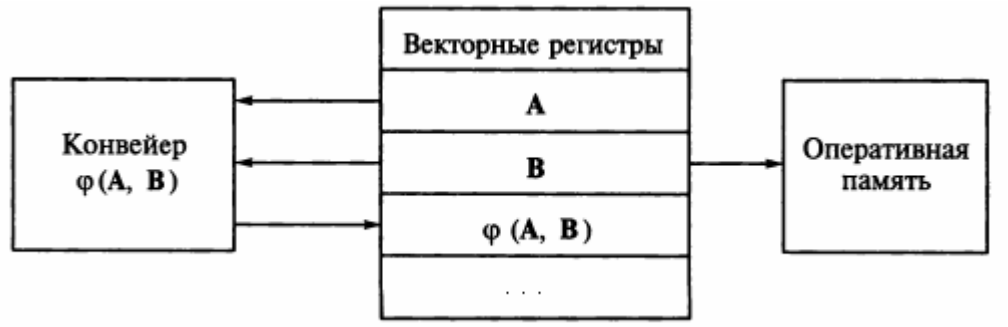
только в унисон, т. е. выполнять одну и ту же векторную операцию (архитектура SIMD). В составе аппаратурно реализованных векторных операций CYBER-205 имелись также триады

$$\boldsymbol{A} + \alpha \boldsymbol{B}$$

Триады выполнялись почти с такой же скоростью, как отыскание суммы или произведения векторов.

6.3. Конвейерные системы типа «регистр-регистр»

Разработкой таких систем занималась фирма Cray Research Inc.



Архитектура такого типа предопределяет в составе ВС векторные регистры, каждый из которых способен хранить вектор-операнд. При реализации векторных команд векторы-операнды извлекаются покомпонентно из векторных регистров, а вектор-результат запоминается также в одном из векторных регистров. До начала реализации векторной команды векторы-операнды должны быть загружены в векторные регистры из оперативной памяти. Предусматривается возможность переноса векторов-результатов из векторных регистров в память.

Система CRAY-1 фирмы Cray research Inc.: функциональная структура и особенности архитектуры

Вычислительная система Cray-1 предназначалась для векторной и скалярной обработки данных. Эта система состояла из четырех функциональных подсистем: управления программой, конвейеров, регистров, памяти и ввода-вывода

Рис. 4.5. Функциональная структура системы Cray-1

Подсистема конвейеров - это и есть процессор ВС Cray-1. Он состоял из 12 функционально ориентированных конвейеров, которые подразделялись на четыре группы: для операции над адресами, скалярных операции, операций над числами с плавающей запятой и векторных операций. Конвейеры состояли из сегментов ЭБО. Каждый ЭБО был ориентирован на выполнение своей микрооперации, длительность цикла любого ЭБО составляла 12,5 нс. Каждый конвейер мог выдавать результаты на каждом цикле работы, следовательно, цикл системы 12,5 нс.

Подсистема регистров ВС Cray-1 включала следующие основные регистры с программным доступом:

1) 8 24-разрядных адресных А-регистров;

2) 64 24-разрядных промежуточных адресных В-регистров;

3) 8 б4-разрядных скалярных S-регистров.,

4) 64 64-разрядных промежуточных скаля рных Т-регистров;

5) 8 векторных V-регистров.

Каждый из V-регистров был способен хранить вектор из 64-х 64-разрядных компонентов. Кроме этих пяти групп регистров имелась также: программно доступный регистр, устанавливавший необходимую длину векторов; 64-разрядный регистр маскирования векторов, разряды которого соответствовали элементам векторных регистров; 64-разрядный регистр часов реального времени.

Подсистема памяти и ввода-вывода ВС Cray-1 имела в своем составе оригинально организованную оперативную память. Последняя обладала емкостью 1 м слов и состояла из 16 независимы: банков емкостью 64 К слов каждый. В свою очередь, любой банк включал в себя 72 модуля памяти, причем каждый из них предназначался для хранения одного разряда всех слов данного банка. Из 72 разрядов слова 64 служили в качестве рабочего слова (команды или операнда), а остальные 8 разрядов предназначались для исправления одиночных и обнаружения двойь ых ошибок в рабочем слове. Время цикла одного банка было равно четырем циклам системы, т. е. составляло 50

нс. Однако наличие 16 независимых банков позволило организовать 16-кратное чередование адресов.

Ввод-вывод информации в ВС Cray-1 осуществлялся через 12 входных и 12 выходных каналов, которые обеспечивали суммарную скорость 500 тыс. 64-разрядных слов в секунду.

Особенность архитектуры ВС Cray-1 состоит в том, что она обладает способностью адаптации к структуре решаемой задачи. Последнее достигается настройкой (программным формированием) цепочек (макроконвейеров) из произвольного числа конвейеров и с произвольной их последовательностью.

В системе одновременно могло выполняться несколько как скалярных, так и конвейерных операций.

Мультиконвейерные системы семейства CRAY: CRAY (X-MP, -2, Y-MP, C90, T932)

Система Cray X-MP это кластер из конвейерных процессоров и относящийся к классу MIMD, в системе может быть 2 или 4 процессора производительностью каждый 235 MFLOPS максимум системы 940 MFLOPS. Область применения Cray X-MP является моделирование авиакосмических объектов. Задачи в этой области допускают расщепление вычислительных процессоров на 2 или 4 самостоятельных процессора.

Cray Y-MP по архитектуре превосходит Cray X-MP, количество процессоров 2 – 8, производительность одного процессора 333 MFLOPS (максимум системы 2,65 GFLOPS). Создана 1988 г.

Максимальное быстродействие Cray C-90 – 16 GFLOPS, количество процессоров 2,4,8,16. Емкость ОП от 512 мб до 8 гб. Класс MIMD. Функциональная структура близка к архитектуре Cray-1. Конвейеры и регистры предназначались для обработки данных 3-х типов: адреса, скаляры и векторные операции.

Конвейерные подразделялись на 4-ре группы: адресные, скалярные, векторные и с плавающей запятой.

Регистры имели дело как с конвейерами, так и с ОП, что удовлетворяет типу регистр-регистр. ОП – общедоступная. Ввод вывод представлялся 3-мя видами типов каналов отличающиеся скорость передачи. В системе реализованы следующие режимы многопроцессорной обработки:

1) Выполнение нескольких различных программ на различных процессорах
2) Выполнение одной параллельной программы на нескольких процессорах

Cray T-90 была продолжением C-90, максимальное быстродействие 64 GFLOPS, промышленные модели данной ВС: Cray T94, Cray T916 и Cray T932, могли состоять из 4,16,32 процессоров, емкость ОП от 512 мб до 8 гб, система имела архитектуру MIMD и была возможность построения макросистем из нескольких T-90.

Cray-2: тип систем PVP, 4-х процессорная конфигурация, производительность 1,95 GFLOPS и 250 MIPS. Архитектура Cray-2 отличалась от Cray-1. Она имела новый набор команд и новую ОС. В них каждому процессору помимо векторных регистров добавлена локальная ОП.

Системы Cray вида PVP (параллельные векторные процессоры) являются коллективами (или кластерами), образованными из конвейерных процессоров

6.4. Конвейерные MIMD-системы

Системы с массовым параллелизмом (MPP-системы)

Система CRAY T3D

Первая 3D система Cray. Количество элементарных процессоров 32 до 2048. Емкость ОП от 512 мб до 128 гб. Производительность от 5 до 300 GFLOPS. Архитектура системы – MIMD, при достаточно полном воплощении принципов коллектива-вычислителей. В системе была обеспечена высокая надежность, живучесть.

Система Cray T-3D работает под управлением Хост машины, основные её функции – это управление программами и данными. В качестве хост-системы могут выступать Cray Y-MP или Cray C-90. Cray T-3D представляет собой композицию из множества вычислительных узлов, коммуникационных сетей, каналов ввода-вывода информации и средства синхронизации. Все вычислительные узлы были однородные, каждый узел включал в себя 2 процессора и коммутатор, локальную память и устройство управления памятью. В качестве процессора использовались DEC 21064α, это RISC процессор с кэш-памятью для команд и кэш-памятью для данных. Набор команд предусматривал операции над целыми и вещественными числами. Локальная память была DRAM памятью емкостью от 16 до 64 мб. Локальный коммутатор обеспечивал связь с соседними узлами и представлял из себя 6-и полюсник. В состав локального коммутатора входит сетевой маршрутизатор, сетевой интерфейс, контролер доступа к памяти. Сетевой маршрутизатор определяет путь перемещения каждого пакета данных и может осуществлять параллельный транзит данных по всем трем межузловым связям. Сетевой интерфейс служит для кодирования информации пред пересылкой и для приема данных для их распределения между ЭП данного ВУ. Контроллер для пересылки блоков данных осуществляет асинхронное перераспределение данных.

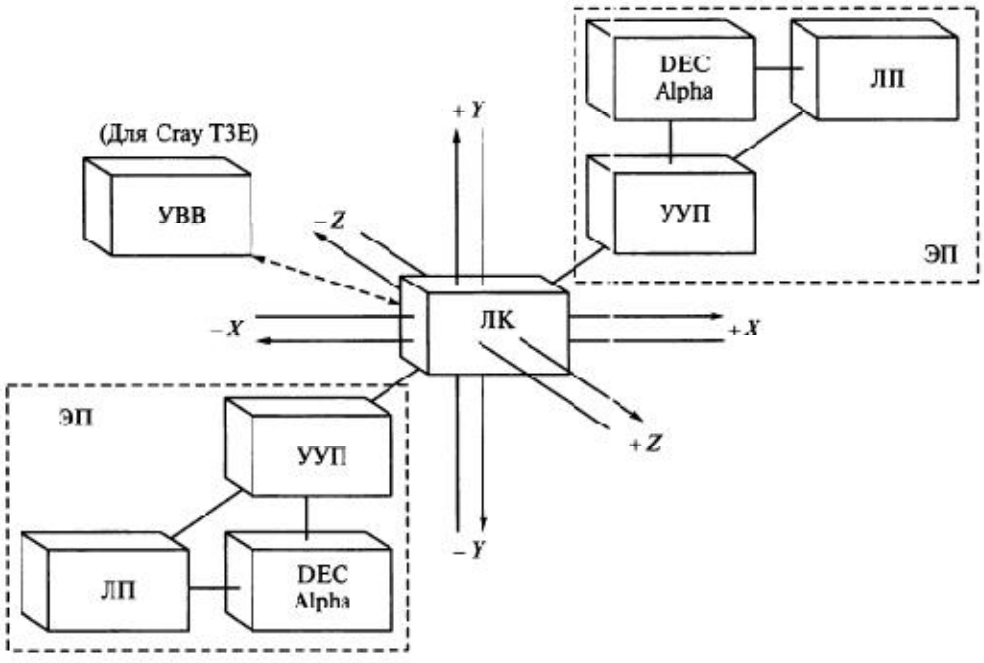


Рис. 4.6. Вычислительный узел системы Cray T3D (Cray T3E):

УВВ — устройство ввода-вывода; ЛП — локальная память; УУП — устройство управления памятью; ЭП — элементарный процессор; ЛК — локальный коммутатор

Коммуникационная сеть Cray T-3D предоставила возможности для реализации администрирование узлов, а также организации взаимодействия между вычислительными

узлами. Сеть ориентирована на решение системных 3D задач и представляла из себя 3D – Тор.

Преимущества 3D-тора:

- Возможность быстрой связи граничных узлов
- Небольшая задержка
- Повышение живучести структур

Адресация вычислительных узлов разделяет на физические, логические и виртуальные. Каждому вычислительному узлу присваивался физический адрес и мог быть присвоен логический адрес, определяющий местоположения логической конфигурации системы. Виртуальная адресация введена для того, чтобы у пользователя был дополнительный сервис, соответственно пользователю при программирование не нужно учитывать логические и физические адреса узлов.

Анализ конвейерных вычислительных систем

Конвейерные процессоры являются пределом модификации архитектуры последовательной ЭВМ. Высокий уровень быстродействия был достигнут в конвейерных ВС за счет мультиконвейерности (параллельной работы множества конвейеров) и конвейеризации на микроуровне.

Параллельно-векторные системы (PVP-systems) отражают предельный вариант в совершенствовании архитектуры конвейерных ВС. В самом деле, PVP-система это коллектив, образованный из процессоров, а последние, в свою очередь, есть ни что иное, как композиция конвейеров.

Разнообразие конвейерных ВС следствие возможностей в технической реализации модели коллектива вычислителей. Три принципа, положенные в основу коллектива вычислителей, достаточно ярко проявляются во всех конвейерных системах.

1. Параллельность выполнения операций.
2. Программируемость структуры
3. Конструктивная однородность

Мультиконвейерные ВС (PVP-systems) позволили достичь быстродействия, измеряемого в десятках GigaFLOPS.

Диалектическое развитие архитектуры конвейерных ВС привело к следующим результатам :

конвейерные ВС переродились в распределенные мультипроцессорные системы с MIMD-архитектурой и массовым параллелизмом;

2) каноническая структура конвейера, которая была основой архитектуры суперЭВМ (конвейерных ВС 1970-1980-х годов), получила внедрение в микропроцессорных больших интегральных схемах

3) современные «конвейерные» микропроцессоры по своей производительности превосходят векторные суперЭВМ 1970-1980-х годов.

Матричные вычислительные системы

7.1. Каноническая функциональная структура матричного процессора

По архитектуре – SIMD. Являются MPP-системами. Данный класс ВС предназначался для решения сложных задач, в которых преобладали операции над векторами и матрицами данных. Первые системы относятся к 1970 году и обладали производительностью 100 млн опер./с, что для того времени позволяло относить их к суперкомпьютерам.

Назначение матричного процессора

Матричные или векторные (Array) процессоры – «матрицы» ЭП, взаимодействующие через сеть связей и работающие под управлением единого устройства.

Матричный процессор предназначенных для решения сложных задач, представленных в параллельной форме. Режимы мультипрограммирования в матричных процессорах не предусматривались. Но, в принципе, режим параллельного мультипрограммирования можно реализовать. Для этого требуется использовать режим «разделения пространства и времени».

SIMD-архитектура

Единое устройство управления, где хранится программа вычислений, множество одинаковых элементарных процессоров. Локальная память предназначена для хранения порций данных. УУ направляет поток команд, которые выполняются одновременно всеми ЭП, но каждый над своими данными.

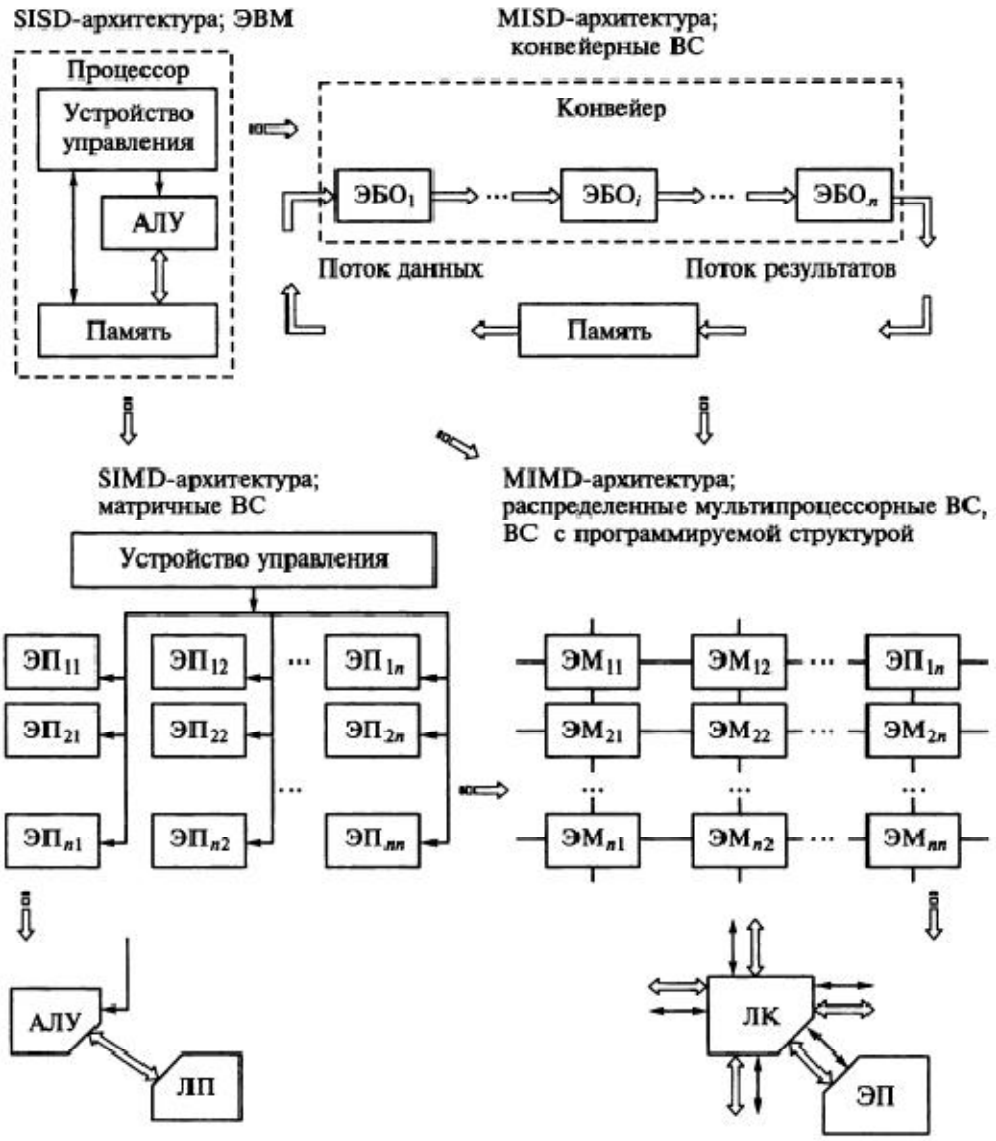


Рис. 3.7. Развитие архитектуры вычислительных средств:

АЛУ — арифметико-логическое устройство; ЭБО — элементарный блок обработки; ЭП — элементарный процессор; ЭМ — элементарная машина; ЛП — локальная память; ЛК — локальный коммутатор; → — поток команд; ⇨ — поток данных; ▫⇨ — направление трансформации архитектуры

Структура и функционирование матричного процессора

В состав ЭП входят АЛУ, ЛП и ЛК. ЛП предназначается для хранения части исходного массива данных. Программа вычислений хранится в УУ – направляет текущую команду во все ЭП, которые параллельно используют команду, но каждый над своими данными. Данная функциональная структура была технико-экономически обоснована в 1960 году.

Режим разделения времени используется в УУ. Его время делится между различными параллельными программами, находящимися в его памяти. Разделение пространства имеет место в матрице ЭП. Для каждой задачи отводится свое подпространство ЭП. Матричный процессор, в отличие от конвейерного, не имеет принципиальных ограничений по наращиванию производительности.

Система SOLOMON (Simultaneous operation Linked Ordinal Modular Network)

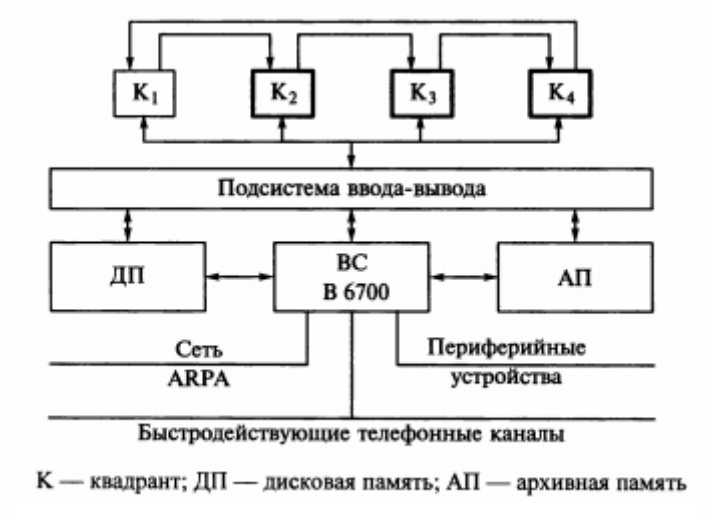
Первая матричная система – система SOLOMON (Simultaneous Operation Linked Ordinal MOdular Network - вычислительная сеть синхронно функционирующих упорядоченных модулей). В системе была каноническая функциональная структура. Планировалось, что она будет представлять собой матрицу 32х32 ЭП, при этом процессоры планировалось использовать одноразрядные. Следовательно, обработка информации была все же последовательной. Закладывалась возможность обработки матрицы чисел с разрядностью от 1 до 128. Емкость локальной памяти – 16 Кбит. Первую пробную версию реализовали с матрицей 3х3, в 1963 г. – 10х10. На этом остановились, но наработки повлияли на дальнейшие разработки. Дальнейшая разработка – ILLIAC IV.

7.2. Система ILLIAC-IV Иллинойского университета и фирмы Burroughs

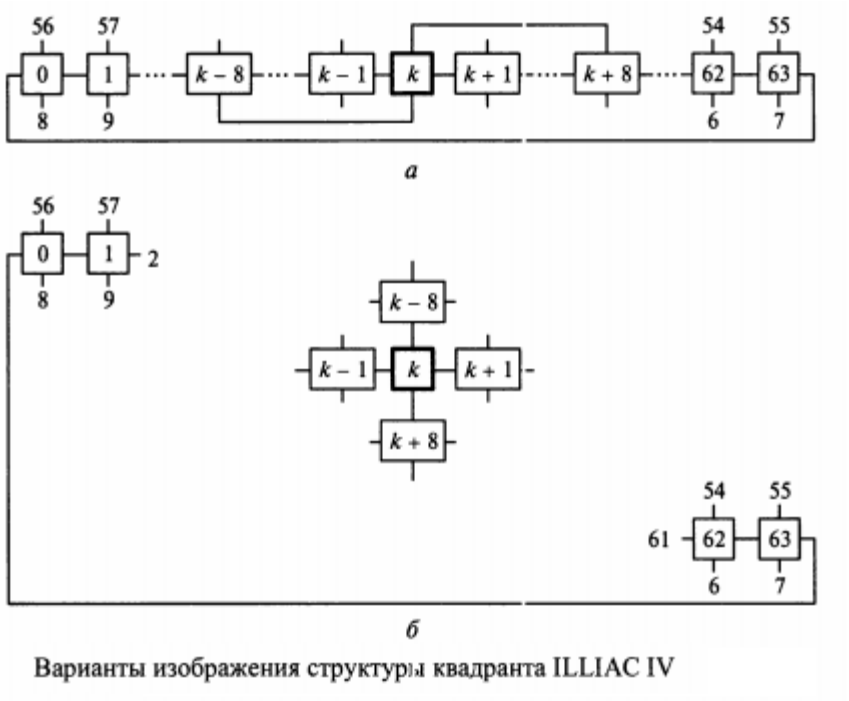
Работы по созданию велись в 1966-1972 гг. В мае 1972 г. фирмой Burroughs собрана в исследовательском центре NASA. По замыслу, планировали млрд опер./с, а на деле получилось 200 млн опер./с над 64-разрядными числами. Вес – 75 тонн, занимаемая площадь – 930 кв. м., стоимость – 40 млн долларов.

В состав входило 64 ЭП. КПД – 80-85%, т. е. столько времени тратилось на решение задач. Соответственно, 15-20% времени тратилось на обслуживание и ремонт. Система включена в состав сети ARPA (Advanced Research Projects Agency). Была самой мощной до 1980 г. и эксплуатировалась до 1981 г.

Функциональная структура системы ILLIAC-IV



К — квадрант; ДП — дисковая память; АП — архивная память



Варианты изображения структуры квадранта ILLIAC IV

Система должна была состоять из 4-х квадрантов, но состояла из одного.

Дисковая память (ДП) состояла из двух дисков и обрамляющих электронных схем.

Архивная память (АП) постоянная лазерная память с однократной записью.

Емкость дисковой памяти – млрд. бит, архивной – трлн. бит. Каждый квадрант – матричный процессор 8х8 – 64 ЭП.

Устройство управления представляло собой специализированную ЭВМ В 6700, которая использовалась для выполнения операций над скалярами и формировала поток команд на матрицу ЭП.

Элементарные процессоры матрицы регулярным образом были связаны друг с другом. Две вершины связаны ребром, если разница между их номерами 1 и 8 по модулю 64.

Структура квадранта системы ILLIAC IV представлялась двумерной решеткой, в которой граничные ЭП были связаны по канонической схеме скалярами и формировала поток команд на матрицу ЭП. (Позднее подобные структуры стали называть D_n-графами, в 90-х эти структуры стали называть циркулярными)

Каждый ЭП имел накапливающий сумматор, регистр второго операнда, регистр передаваемой информации, регистр, использовавшийся как временная память, регистр модификации адресного поля команды, регистр состояния данного ЭП.

Подсистема ввода-вывода состояла из устройства управления, буферного запоминающего устройства и коммутатора. Комплекс этих устройств обеспечивал обмен информацией между квадрантами ILLIAC N и средствами ввода-вывода: ЭВМ В 6700, дисковой и архивной памятью, периферийными устройствами

Архитектурные возможности квадранта и элементарного процессора

Квадрант – матричный процессор, включавший в себя устройство управления и 64 ЭП.

Матрица из 64 ЭП предназначалась для реализации операций над векторами. Числа могли быть с плавающей запятой разрядностью 32 или 64 бита. Производительность зависела от разрядности чисел. Память каждого ЭП состояла из 2048 64-разрядных слов.

Элементарный процессор мог находиться в одном из двух состояний – активном или пассивном. В первом ему разрешалось, а во втором запрещалось выполнять команды, поступавшие из устройства управления. Состояние ЭП задавалось при помощи специальных команд.

7.3. Система DAP (Distributed Array Processor) фирмы ICL (International Computer Ltd.)

Особенности архитектуры, структуры сети межпроцессорных связей и элементарного процессора

SIMD-система, разработанная фирмой ICL (International Computers Ltd.).

Планировалось 50000 ЭП. Но на практике: 1972 г. – 32x32 – 1024 ЭП, 1977 г. – 64x64 – 4096 ЭП.

Функциональная структура ВС DAP - это композиция ведущей ВС и собственно DAP. Ведущая ВС Host Computer) предназначалась для реализации функций операционной системы (включая подготовку данных и команд для DAP, распределение дам оных по ЭП). Матричная система DAP применялась для массовых параллельных вычислений.

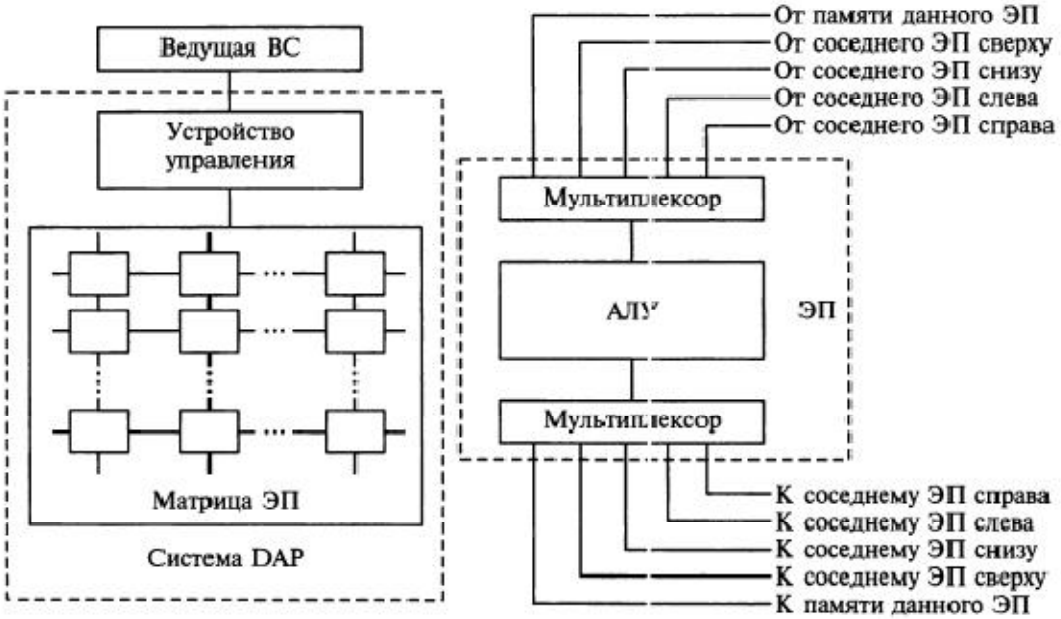


Рис. 5.4. Функциональная структура ВС DAP:
ЭП — элементарный процессор; АЛУ — арифметико-логическое устройство

Устройство управления формировало поток команд на матрицу ЭП, в частности оно направляло команды, адреса и другую информацию, необходимую элементарным процессорам для выполнения «матричных» операции.

Каждый элементарный процессор DAP представлял собой одноразрядный микропроцессор, связанный с локальной памятью емкостью в 4096 бит. Матрица ЭП и их память для удобства представлялись в виде «прямоугольного вычислительного параллелепипеда» из 4097 горизонтальных слоев. Верхний слой параллелепипеда это матрица ЭП, в которой строка имела длину Ld элементов (d - длина слова ведущей ВС, L - число слов ведущей ВС в строке), а столбец состоял из 2^m ведущей ВС, L число слов ведущей ВС в строке), а столбец состоял из 2" элементов (т некоторое выбранное целое положительное число). Она была способна параллельно обрабатывать $L*2^m$ d-разрядных

слов. Локальная память всех ЭП составляла распределенную память системы DAP в целом.

Между ЭП существовала сеть связей. Она обеспечивала через мультиплексоры связь каждого ЭП с регистрами АЛУ четырех ближайших соседей, расположенных сверху, снизу, слева и справа от него.

Каждый ЭП имел в своем составе предельно простое одноразрядное АЛУ, которое обеспечивало последовательную поразрядную обработку информации.

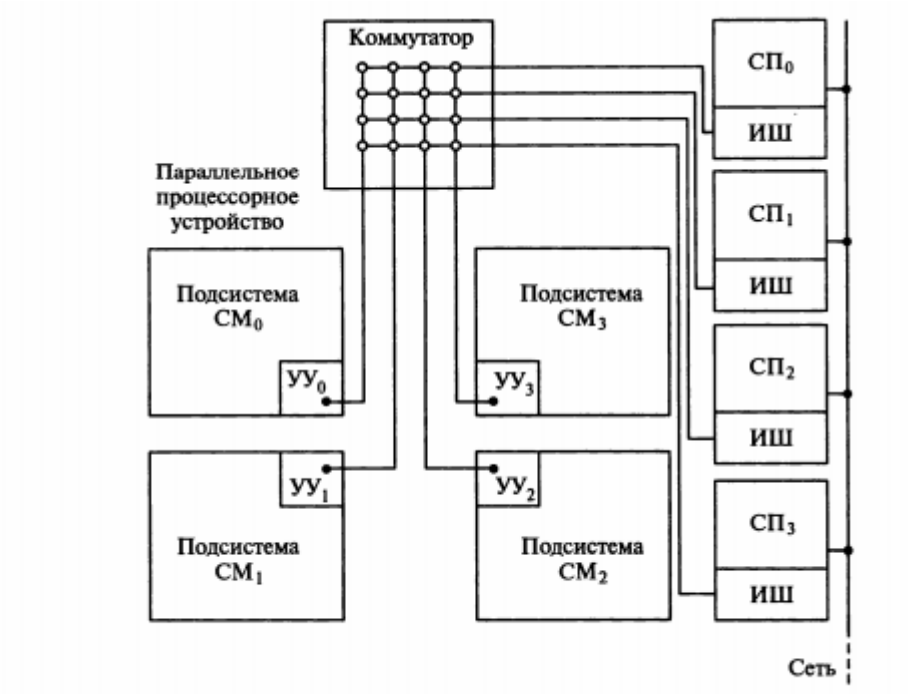
В каждом ЭП был заложен механизм, позволявший программировать состояние процессора. Этот механизм был предельно прост и представлял собой одноразрядный регистр активности, одно из состояний которого (пассивное состояние) запрещало ЭП выполнять поступавшую команду.

Команды для матрицы ЭП DAP поступали из устройства управления. Они содержали всю информацию, которая требовалась для выполнения операции.

Архитектурной особенностью ВС DAP . являлось и то, что ее распределенная память могла быть использована вед. щей ВС CDC 7600 как обычная память. Еще одна особенность системы: каждый ЭП обладал архитектурой SISD.

7.4. Семейство систем Connection Machine (CM) фирмы Thinking Machine Corp.

Функциональная структура систем семейства CM (подсистемы CM^i, $i \in \{0,1,2,3\}$, матричный коммутатор, коммуникационные процессоры)



СП — сервисный процессор; ИШ — интерфейс шин; УУ — устройство управления

В состав ВС входили: 4 СП (КП – коммуникационный процессор) с ИШ, коммутатор и параллельное процессорное устройство, состоящее из четырех подсистем СМ0-СМ3. Каждая подсистема имеет в своем составе УУ и 16384 ЭП.

УУ – специально спроектированный микрокомпьютер. Применяется для реализации функций виртуальной машины, т. е. программирует виртуальную систему. Это устройство содержит память нанокоманд емкостью 16К 96-разрядных слов. На входы четырех УУ поступает поток команд высокого уровня, т. е. операции виртуальной машины и аргументы. Этот поток поступает из коммутатора по параллельному 32-

разрядному каналу. На выходе УУ имеет место поток нанокоманд, которые управляют ЭП.

Основной элемент в составе подсистемы – вычислительный узел, в состав которого входит 16 ЭП, память и маршрутизатор. Каждый узел – композиция из двух кристаллов, один кристалл – памяти, другой – маршрутизатор и 16 ЭП. Маршрутизатор – УУ для ВУ. ЭП связаны в решетку. Т. о. архитектура ВУ – SIMD. В состав каждого ЭП входят одноразрядное АЛУ, битно-адресуемая локальная память емкостью 4 Кбит, 8 одноразрядных регистров-признаков (флагов) и интерфейс маршрутизатора и двумерный интерфейс сети межпроцессорных связей, т. е. архитектура ЭП - SISD. ВУ в пределах любой подсистемы образуют 10D-куб. Т. о. архитектура подсистемы – SIMD, но сети связей между узлами не решетка, а гиперкуб.

КП, по сути, составляют аппаратурно-программную среду для разработки параллельных программ и выполняют функции хост-компьютера.

ИШ поддерживает 32-разрядный параллельный канал

Коммутатор обеспечивает связь любого КП с любой из подсистем или их подмножеством.

Работа системы в целом может быть организована так, что один КП работает с приписанной ему подсистемой. Следовательно, архитектура системы в целом – MIMD. В этом суть мультиархитектуры, при этом каждый ЭП может находиться в активном или пассивном состоянии.

Элементарные процессоры систем СМ

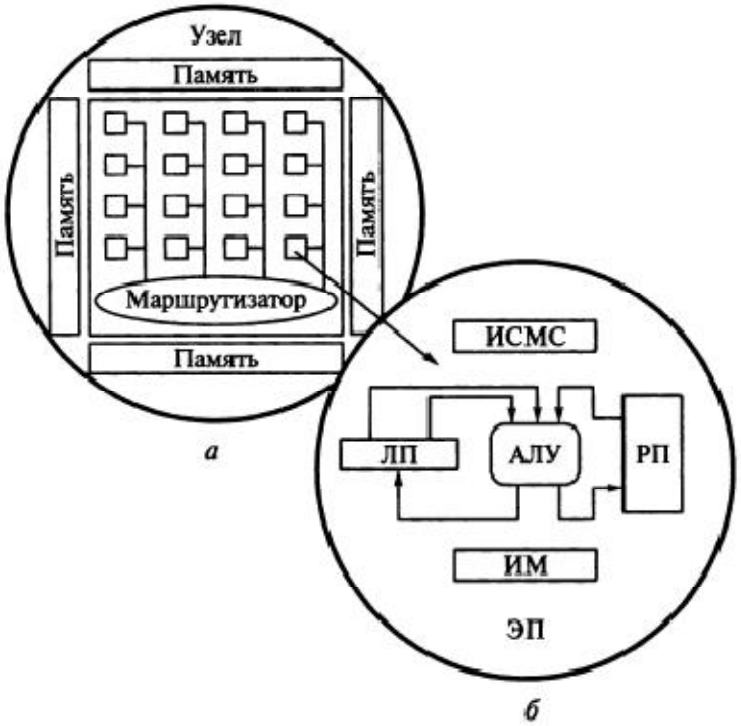


Рис. 5.6. Функциональные структуры ЭП и узла ВС СМ-1:

а — ВУ; *б* — ЭП; ИСМС — интерфейс сети межпроцессорных связей; ЛП — локальная память; АЛУ — арифметико-логическое устройство; РП — регистр признаков; ИМ — интерфейс маршрутизатора; ЭП — элементарный процессор

Он имеет архитектуру SISD и является одноразрядным последовательным средством обработки информации. В состав каждого ЭП входят:

• одноразрядное АЛУ;

• битно-адресуемая локальная память (ЛП) емкостью 4 K бит;
• восемь одноразрядных регистров признаков (РП) или флагов;
• интерфейс маршрутизатора (ИМ);
• двумерный интерфейс сети межпроцесссрных связей (ИСМС).

Арифметико-логическое устройство ЭП имеет три входа и два выхода и включает в себя логические элементы, одноразрядные регистры-защелки (Latches) и интерфейс памяти

Сеть микропроцессорных связей систем СМ

Поддерживаются следующие механизмы.

Широковещательная управляющая сеть

Обеспечивает в ВС безотлагательный прием одновременно всеми ЭП управляющей информации, поступающей от устрой яства управления или коммуникационного процессора.

Связь «обобщенное ИЛИ»

Осуществляет реализацию логического ИЛИ над значениями переносов в АЛУ всех элементарных процессоров;

Гиперкубическая сеть

Является средой для маршрутизаторов и многочисленных параллельных примитивов, поддерживаемых моделью виртуальной машины.

Маршрутизатор

Непосредственно осуществляет пересылку пакетов сообщений в ЭП.

Межпроцессорная сеть

Предоставляет прямой доступ из данного ЭП к ближайшим соседям в пределах узла

Модель виртуальной машины семейства СМ

Архитектура этой машины весьма близка к архитектуре физической системы Connection Machine и имеет два существенных расширения: ее набор параллельных команд (названный Paris - Парис) существенно расширен и в ней имеется абстракция виртуального процессора. Большая часть набора Paris реализова в аппаратуре УУ, где осуществляется синтаксический анализ потока кодов операции и аргументов и его преобразование в соответствующую последовательность нанокоманд для ЭП.

Виртуальный процессор необходим во многих областях параллельной обработки данных (так как часто требуются специфические процессоры, которые заметно отличаются от физических ЭП данной системы).

Отношение V / P = k называют коэффициентом виртуального процессора

P - реальное число физических ЭП

V - виртуальных процессоров

Программное обеспечение систем семейства СМ

Операционная система (Operating System), являющаяся штатной операционной средой (либо UNIX, либо LISP) сервисных процессоров с небольшим расширением. Пользователям предоставляется возможность применять языки и конструкции всего программного инструментария сервисных процессоров. Кроме того, пользователи могут

без особого труда разработать программы, рассчитанные на эксплуатацию всей вычислительной мощности аппаратуры системы Connection Machine.

Языки CM-FORTRAN, Star LISP, CM-LISP, C*

Модели семейства CM

CM-1

Разработка велась в 1983-1986 гг. Максимальное быстродействие – 2000 MIPS над 32-разрядными словами. Емкость ОП – 32 МБ.

CM-2

Совместима с предшествующими моделями и обладала большими вычислительными возможностями. В ней аппаратурно реализуются операции с плавающей запятой (1986-1987). Максимальное количество ЭП – 64К. Быстродействие 2500 MIPS или 32 GFLOPS.

Функциональная структура получила развитие: введены подсистемы ввода-вывода и дисковая память большей емкости. Функциональная структура стала более масштабируемая и с минимальной конфигурацией из двух вычислительных узлов (32 ЭП). В ЭП добавили ускоритель для операций с плавающей запятой и увеличили емкость памяти до 64 Кбит.

CM-5 (1991)

16384 ЭП, производительность – 1 TFLOPS. Емкость памяти – 512 ГБ. Получила развитие архитектура, которая также является композицией из SIMD и MIMD архитектуры. Каждая система масштабируема, варьирование числа ВУ от 16 до 16К. Межузловые связи – от 4D до 14D-куба. Множество ВУ разделено на управляющие (control) и вычислительные(computational). Допускалось создание произвольного количества подсистем, в каждой из которых был управляющий процессор и множество вычислительных.

Анализ архитектуры систем (на макроуровне, в пределах подсистемы CM^i в целом и её вершины, на микроуровне – на уровне элементарного процессора)

7.5. Анализ матричных вычислительных систем

Матричный способ обработки информации в отличие от конвейерного в принципе позволяет осуществлять неограниченное количество вычислительных процессов

- В таких системах в высокой степени воплощены фундаментальные архитектурные принципы. Параллельность выполнения операций в матричных ВС обеспечивается на нескольких функциональных уровнях. На макроуровне параллельность достигается за счет одновременной работы нескольких матричных процессоров. На микроуровне параллельность выражается в возможности одновременной работы большого количества элементарных процессоров.
- Программируемость структуры в матричных системах изначально проявлялась более сильно, чем в конвейерных. Матричная ВС может быть так настроена, что ее различные квадранты или подсистемы будут одновременно решать различные задачи. В пределах квадранта или подсистемы имеется возможность программировать направление передачи информации от каждого ЭП. В матричных

ВС заложены средства программного управления состоянием каждого ЭП. На различных группах ЭП можно выполнять различные программы. В матричной ВС - разделение «пространства» ЭП и системного времени. Матричные ВС располагают функционально гибкой структурой сети межпроцессорных связей

- программируемые структуры сетей межпроцессорных связей позволяют ВС адаптироваться под области применения и структуры решаемых задач, они делают локальную оперативную память любого ЭП общедоступной для других ЭП системы.
- Однородность состава и структуры ВС видна на всех функциональных уровнях. На макроуровне однородность выражена тем, что все матричные процессоры и устройства управления, входящие в них, одинаковы. На микроуровне однородность ВС достигнута за счет применения множества идентичных элементарных процессоров.
- Архитектура матричных ВС не лишена существенны недостатков. Так, единственное устройство управления в квадранте или подсистеме резко снижает надежность и живучесть, а также производительность при мультипрограммной работе и, следовательно, ограничивает сферу применен я матричных ВС.

Мультипроцессорные вычислительные системы

8.1. Каноническая функциональная структура мультипроцессора

MIMD-архитектура

4) MIMD

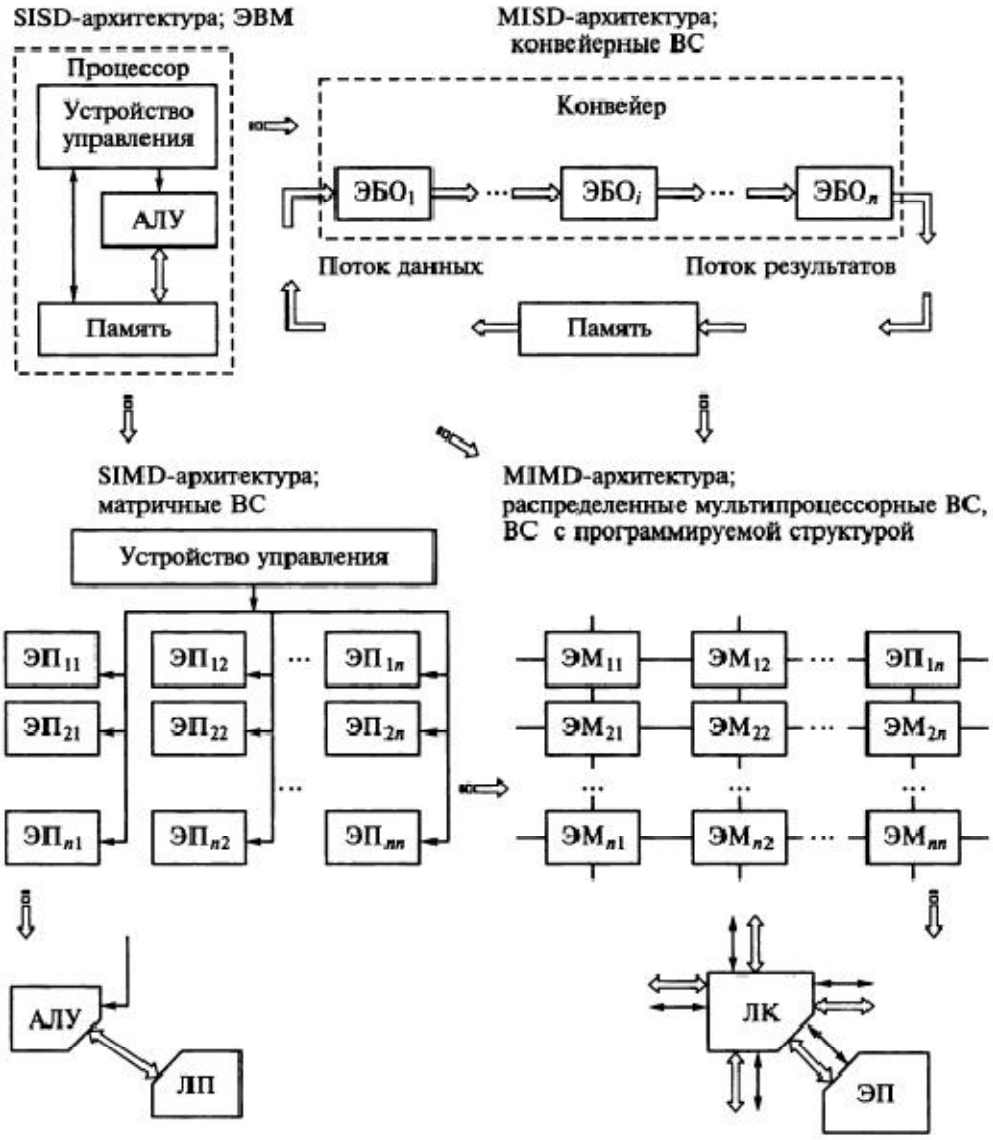


Рис. 3.7. Развитие архитектуры вычислительных средств:

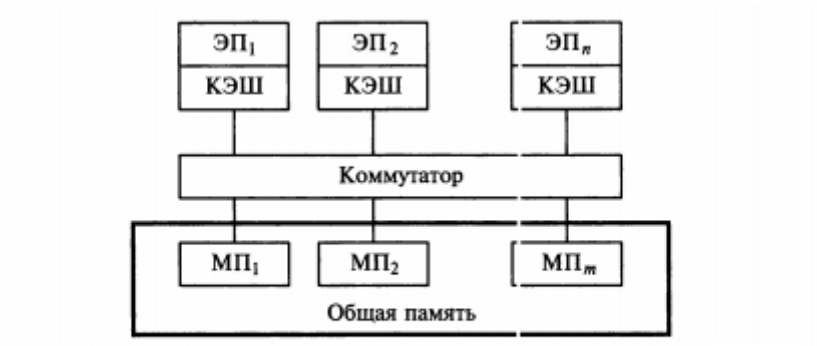
АЛУ — арифметико-логическое устройство; ЭБО — элементарный блок обработки; ЭП — элементарный процессор; ЭМ — элементарная машина; ЛП — локальная память; ЛК — локальный коммутатор; → — поток команд; ⇨ — поток данных; ▫⇨ — направление трансформации архитектуры

Множество элементарных машин, связанных между собой через сеть межмашинных линий. Каждая ЭМ – композиция локального коммутатора и элементарного процессора. Локальный коммутатор обеспечивает связь с соседними ЛК. ЭП имеет полноценное УУ. Его локальная память предназначена для хранения не только части данных, но и ветви параллельной программы. В архитектуре этого класса имеются следующие виды систем:

мультипроцессорные, распределенные, с программируемой структурой, кластерные.

Функционирование мультипроцессора

Функциональная структура основывается на MIMD архитектуре и едином общедоступном ресурсе. Т. о., по определению, мультипроцессорная ВС – совокупность процессоров, взаимодействующих между собою через единый общедоступный ресурс. В качестве таких ресурсов выступает общая ОП, общее внешнее ЗУ, ЭВМ-посредник или коммутатор общей шины.



Каноническая функциональная структура мультипроцессора:
ЭП — элементарный процессор; МП — модуль памяти; КЭШ — кэш-память

Мультипроцессор – композиция из множества ЭП, МП и коммутаторов, обеспечивающих взаимодействие между элементами двух подмножеств. Любая область любого модуля памяти доступна любому ЭП. Следовательно, множество МП образует общедоступную память. Коммутатор может быть как сосредоточенным, так и распределенным. В последнем случае он представляется композицией ЛК, каждый из которых находится во взаимно однозначном соответствии либо с ЭП, либо с МП. Коммутатор в данном случае не является общедоступным ресурсом. Обмен информацией между ЭП осуществляется не через коммутатор, а через общую память. В функциональной структуре вместо коммутатора может использоваться шина или система шин. В теоретических исследованиях по изучению параллельных алгоритмов используется идеализированный мультипроцессор PRAM(Parallel Random Access Machine – параллельная машина с произвольным доступом к памяти). В PRAM-модели предполагается, что время обращения любого ЭП к любой ячейке любого МП – постоянная величина. Поэтому необходимость масштабирования мультипроцессора однозначно определяет иерархию памяти. Наряду с ОП вводится сверхбыстрая память – КЭШ. Она работает с большей скоростью, чем ОП. В нее заносятся копии часто используемых данных, поэтому она сокращает число обращений к ОП. Кэш может быть сосредоточенный, либо распределенный по ЭП.

ВС, основанные на канонической функциональной структуре, относятся к системам с общей (разделяемой, true shared) памятью. В первых мультипроцессорных ВС количество процессоров было около 10, в современных суперВС – сотни тысяч (MPP).

8.2. Система C.mmp (Carnegie-Mellon Multi-Mini-Processor) Университета Карнеги-Меллона

Вычислительная система C.mmp (Carnegie-Mellon Multi-Processor) была создана в университете Карнеги-Меллона (США).

Функциональная структура мини-ВС C.mmp

Рис. 6.2. Функциональная структура системы C.mmp:
ЭП — элементарный процессор; МП — модуль памяти

Систем состояла из 16 ЭП, общей памяти и матричного коммутатора. В состав ЭП (мини-ЭВМ корпорации DEC, Digital Equipment Corp.) входили незначительно модифицированный процессор PDP-11 /40, локальная (или местная, или индивидуальная) память, блок отображения адреса (который преобразовывал генерировавшиеся процессором 18-разрядные адреса в 25-разрядны. физические адреса) и контроллер межпроцессорного интерфейса (который обеспечивал подключение процессора к межпроцессорной шине). Кроме указанных компонентов в состав ЭП могли входить память на магнитных дисках, страничная память на дисках, внешние устройства и др. Матричный коммутатор 16 x 16 позволял установить связь между любым процессором ЭП и любым портом МП; (i, j €{1, 2, ..., 16}) памяти общего доступа.

Анализ надежности мини-ВС C.mmp

При использовании такой модели ВС считается, что вероятность безотказной работы в течении времени t любой подсистемы из N идентичных модулей равна

$$R(t) = \sum_{i=0}^{N-n} \frac{N!}{i!\,(n-i)!} r^{N-i}(t)[1-r(t)]^i$$

Где r(t) = exp(-λt) – функция надежности
λ – интенсивность отказов модуля
n – допустимое число исправных модулей
(N – n) – резерв

Анализ надежности ВС C.mmp показал, что сосредоточенный коммутатор является критическим источником отказов в мини-ВС.

Система C.mmp и при отсутствии средств восстановления обладала надежностью, допускавшей решение сложных задач.

Недостатки архитектуры мини-ВС C.mmp

- Сосредоточенный коммутатор в мини-ВС C.mpp являлся критическим источником отказов. Как показали исследования, вероятность безотказной работы системы резко убывает на начальном участке времени.

- Существуют границы для количества резервных модулей. Увеличение количества резервных модулей сверх этих границ практически не улучшает надежности систем.

- Заметное повышение надежности мини-ВС обеспечивается применением высоконадежных компонентов. Однако такой путь имеет свои физические и технические пределы.

- Кардинальный путь повышения надежности ВС состоит не в увеличении надежности компонентов, а в применении перспективных архитектурных и структурных решений, новых принципов обработки информации

Принципиальным недостатком структуры мини -ВС C.mpp является матричный коммутатор, выход которого из строя приводит к отказу системы как единого ансамбля модулей

8.3. Вычислительные системы семейства Burroughs

Фирма «Бэрроиз» (Burroughs Corp.) в 1961 г. начала работы по созданию своего семейства многопроцессорных ВС. В 1963 г. была выпущена первая ВС В 5000, а в 1964 г. ее модификация В 5500; в 1969 г. была создана В 6500, в 1971 г. В 6700 и, наконец, в 1973 г. В 7700.

Системы В 6700 и 7700

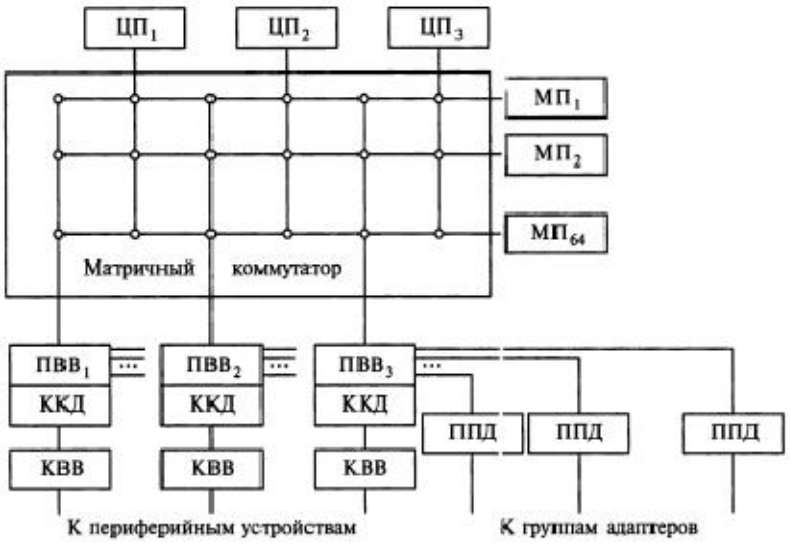


Рис. 6.3. Функциональная структура ВС В 6700:

ЦП — центральный процессор; ПВВ — процессор ввода-вывода; ККД — канал коммутации данных; КВВ — контроллер ввода-вывода; ППД — процессор передачи данных; МП — модуль памяти

Вычислительная система В 6700 (1971 г.) — это композиция ЭП, модулей памяти, коммутатора и периферийного оборудования. Подмножество ЭП составляли 1-3 центральных процессоров и 1-3 процессоров ввода-вывода.

Центральный процессор (ЦП) системы В 6700 обладал быстродействием порядка 1 млн опер./с (над 48-разрядными числами). Оперативная память состояла из 1-64 модулей (МП), обладала емкостью до 6 Мбайт и имела шесть входов. Для формирования памяти использовались модули емкостью 6 К или 64 К слов. Скорость обмена информацией между процессорами (центральными и/или ввода-вывода) и оперативной памятью по любому из шести входов составляла 6750 К байт/с.

Процессоры ввода-вывода (ПВВ) предназначались для подключения периферийного оборудования, в каждом из процессоров имелось 4-12 каналов коммутации данных (ККД). Три процессора ввода-вывода могли одновременно выполнять до 36 операций ввода- вывода. К каждому каналу могло подключаться до 20 контроллеров ввода-вывода (КВВ), а к каждому контроллеру до 10 периферийных устройств. Максимальное количество адресуемых периферийных устройств составляло 128.

Каждый ПВВ был соединен с 1-4 процессорами передачи данных (ППД), каждый из которых был связан с 1-16 группами адаптеров, а каждая из них, в свою очередь, соединялась 1-16 линиями связи.

Вычислительная система В 7700 (1973 г.) в отличие от В 6700 могла иметь в своем составе 1-7 ЦП и от 1-7 процессоров ввода-вывода информации, при этом общее количество процессоров не превышало восьми. Быстродействие ЦП составляло 4...5 млн опер./с.

Процессор ввода-вывода обслуживал 32 канала и четыре процессора передачи данных. К каждому каналу подключалось периферийное оборудование; максимальное количество адресуемых внешних устройств составляло 255.

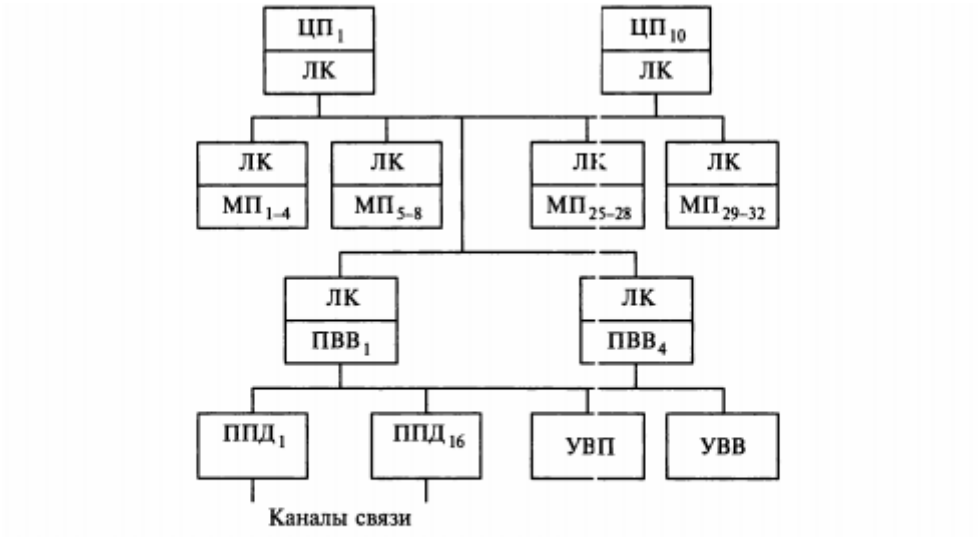
Процессор передачи данных был рассчитан на подключение до 256 линий связи. Оперативная память ВС В 7700 состояла из восьми модулей, была 8-входовой и имела емкость 12 288 К байт. Слово состояло из 52 разрядов, из которых 48 были информационными, 3 управляющими и 1 для контроля по четности. Все разряды слова были доступны процессорам, как центральным, так и ввода-вывода и передачи данных.

Программное обеспечение системы В 7700 представляло собой модификацию ПО В 6700. Программное обеспечение в комплексе с аппаратурой выполняло функции обнаружения, локализации и устранения неисправностей и ошибок. При наличии отказов осуществлялось динамическое изменение рабочей конфигурации ВС. Операционная система после обнаружения и локализации отказа или ошибки исключала из рабочей конфигурации неисправные модули или программы с ошибками (без влияния на другие программы).

8.4. Вычислительные системы семейства «Эльбрус» Института точной механики и вычислительной техники им. С.А. Лебедева

Разработана в ИТМиВТ академии наук СССР в 1970-1985 гг. Эльбрус-1 применялись в 1970-80 гг. Эльбрус-2 созданы в 1985 г. и еще 15 лет выпускались серийно, как правило, для военных целей.

Функциональная структура систем семейства «Эльбрус»



Функциональная структура системы «Эльбрус»:

ЦП — центральный процессор; ЛК — локальный коммутатор; МП — модуль памяти; ПВВ — процессор ввода-вывода; ППД — процессор передачи данных; УВП — устройство внешней памяти; УВВ — устройство ввода-вывода

В состав входило до 10 ЦП, от 4 до 32 МП, от 1 до 4 ПВВ, от 1 до 16 ППД, УВВ и УВП. Взаимодействие ЦП, МП и ПВВ осуществлялось с помощью композиций ЛК, представляющих распределенный коммутатор. ППД обеспечивал удаленный доступ к машине. Все возможные внешние устройства могли быть подключены как непосредственно к ПВВ, так и ППД.

Архитектурные особенности

Каждый из элементов ЦП, ПВВ, МП и ЛК имели полный аппаратный контроль. Даже при появлении одиночной ошибки возникал сигнал неисправности, который воспринимался ОС. ОС исключала отказавший элемент из рабочей конфигурации и переводила его в резервную конфигурацию. Для устранения неисправностей в элементе использовался комплекс контрольно-диагностических тестов и специальная аппаратура. После восстановления элемент снова включался в рабочую конфигурацию. Так проявлялась программируемость структуры ВС. Кроме того, предполагалась возможность построения сверхнадежных конфигураций, когда для элемента программировались резервные подобные элементы. В такой структуре в случае отказа элемента его функции могут быть переданы резервному элементу за долю секунды.

Модели семейства «Эльбрус»

Особенности системы «Эльбрус-1»

Первый образец – 1978 г., промышленное производство освоена в 1980-м году. Производительность – от 1,5 до 15 MFLOPS. Емкость памяти – до 4 МБ. Разрядность слова – 32, 64, 128. В состав ПО входила распределенная ОС, система программирования, комплекс прикладных и сервисных программ, системы телеобработки (удаленного доступа) и комплекс программ технического обслуживания. ОС могла работать в режиме пакетной обработки, а также разделяемого времени и дистанционной обработки. ОС осуществляла также динамическое распределение ресурсов ВС, управляла работой процессоров и обеспечивала их синхронизацию. ОС также обеспечивала автоматическую реконфигурацию, восстановление файлов и перезапуск задач в системе. Система программирования представлена всеми распространенными языками на тот момент.

Особенности системы Эльбрус-2

У этой системы была максимальная производительность – 125 MFLOPS. Емкость памяти – до 160 МБ. Пропускная способность распределенного коммутатора – 2 Гбайт/с. Часто встречаемые программные конструкции языков высокого уровня имели аппаратную поддержку. Диспетчеры ПВВ также были реализованы в аппаратуре. Система команд и программное обеспечение ППД обеспечивали простую адаптацию к различным вычислительным сетям и внешним объектам. В этой машине допускалось формирование конфигураций с использованием вместо ЦП специально спроектированного процессора, полностью совместимого с БЭСМ-6.

О других моделях «Эльбрус»

1995 г. – Эльбрус-3 в единственном экземпляре с 16 процессорами. Далее развитие семейства связано с созданием микропроцессоров.

Перспективы развития семейства ВС «Эльбрус»

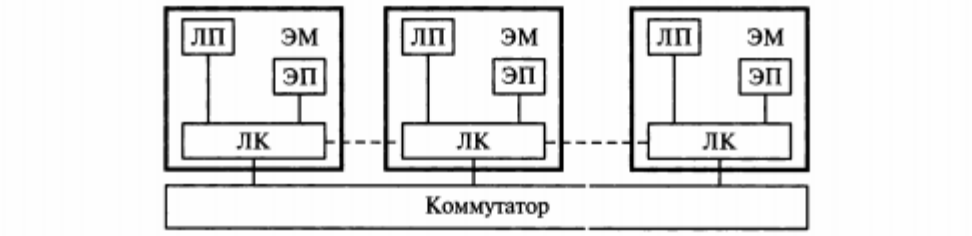
Дальнейшее архитектурное развитие семейства ВС «Эльбрус» было связано с проектированием микропроцессора E2k (E2k сокращение от Elbrus 2000)

Институтом микропроцессорных вычислительных систем РАН были выполнены работы в рамках госзаказа по созданию ВС «Эльбрус-3М»

Сейчас МЦСТ ЭЛЬБРУС (Российские микропроцессоры и вычислительные комплексы) разрабатывают: Восьмиядерный микропроцессор с архитектурой Эльбрус (Эльбрус-8С)

8.5. Предпосылки совершенствования архитектуры мультипроцессорных вычислительных систем

Мультипроцессорные ВС – результат развития архитектуры ЭВМ. В мультипроцессорных ВС принципы модели коллектива вычислителей реализованы далеко не с исчерпывающей полнотой. Причиной этого является единый ресурс, через который взаимодействуют средства обработки и хранения информации. Единый ресурс ограничивает возможности масштабирования, наращивания производительности и емкости памяти, снижает надежность ВС в целом.



Модифицированная функциональная структура мультипроцессора:
ЛП — локальная память; ЭМ — элементарная машина; ЛК — локальный коммутатор; ЭП — элементарный процессор

Необходимость достижения большей масштабируемости и надежности мультипроцессора, чем у канонической структуры, заставила разработчиков перейти к элементам обработки, представленным композицией из ЭП, ЛП, ЛК (иными словами Элементарной машиной(ЭМ)) и организовать их взаимодействие через общий коммутатор. Следующим шагом развития стало использование не сосредоточенного, а распределенного коммутатора. Первым шагом для этого пути является создание

коммутатора на основе принципа модульности. В этом случае ЛК должно быть столько же, сколько и ЭМ. Следующий шаг – расширение функций ЛК в пределах одной ЭМ и организация связей, представленная пунктиром на рисунке. Но эта архитектура относится к полностью распределенным ВС. Еще дальнейшее развитие – превращение одномерной структуры в многомерную, а ЛК – в многопользовательский. ВС на основе модифицированного мультипроцессора называют мультипроцессорными ВС с разделяемой виртуальной памятью (Virtual Shared Memory). Современные мультипроцессоры – MPP-системы, одним из которых является ВС IBM Blue Gene.

8.6. Система Cm Университета Карнеги-Меллона*

*Архитектура микроВС Cm**

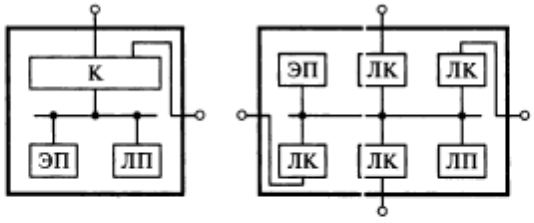


Рис. 6.6. Элементарная машина системы Cm*:

а — на базе контроллера; *б* — на базе локального коммутатора; К — контроллер; ЭП — элементарный процессор; ЛК — локальный коммутатор; ЛП — локальная память

Рис. 6.7. Каноническая структура системы Cm*:

ЭМ — элементарная машина

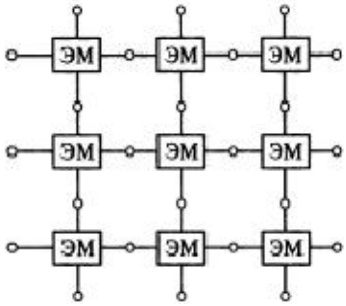


Рис. 6.8. Фрагмент двумерной структуры системы Cm*:

ЭМ — элементарная машина

В микроВС Cm* пара «элементарный процессор локальная память» выполняла функции вычислительного модуля, а вычислительный модуль в совокупности либо с контроллером К отображения адресов (рис. 6,6, а), либо с локальным коммутатором ЛК (рис. 6.6, б) реализовывали функции ЭМ.

Взаимодействие между вычислительными модулями осуществлялось через «распределенный коммутатор» сеть связи, образуемую подсоединением контроллеров отображения адресов к межмодульным шинам (см. рис. 6.7) или (и) отождествлением полюсов локальных коммутаторов различных элементарных машин (см. рис. 6.8)

Контроллер - это специальный процессор, который выполнял все функции, связанные с передачей сообщений.

Архитектура ВС Cm*

1. Реконфигурируемость - способность микроВС к априорной адаптации своего состава и структуры сети межмашинных связей к конкретной области применения
2. Масштабируема - способность микроВС Cm* к развитию в целях увеличения производительности, объема памяти и полосы пропускания каналов связи

3. Общедоступность и распределенность памяти. Память системы Cm* состояла из общей памяти и локальной памяти элементарных машин. Вся память системы была потенциально доступна для всех процессоров.
4. эффективность использования ресурсов системы обеспечивалась возможностью организации параллельной обработки исходных данных элементарными машинами и параллельных межмашинньх обменов информацией.
5. Надежность функционирования микроВС достигалась за счет распределенности ее ресурсов.

*Средства обеспечения надежности микроВС Cm**

В микроВС Cm* реализован подход, основанный на введении в контроллер отображения адреса специальной аппаратуры ловушек (Hooks). Ловушка предоставляла микропроцессору возможность тщательного исследования и изменения внутреннего состояния контроллера.

В локальный коммутатор были введены регистры, которые предназначались для хранения информации при диагностировании и восстановлении после обнаружения программного и аппаратурного отказов.

Все внешние обращения к памяти любого вычислительного модуля выполнялись с помощью коммутации сообщений.

Вся память системы была защищена контролем на четность. При нарушении четности адрес блокировался и поступал соответствующий сигнал в контроллер, с тем чтобы он мог повторить обращение к памяти.

*Система самодиагностики микроВС Cm**

Система самодиагностики размещалась в машине-диспетчере на магнитной ленте

Диагностические программы загружались поочередно; условием загрузки очередной программы в свободный от работы вычислительный модуль было либо успешное завершение заданного числа прогонов, либо обнаружение отказов после определенного количества прогонов текущей диагностической программы.

Система самодиагностики была способна реагировать на некоторые ситуации автоматически. Одна из таких ситуаций зацикливание модуля.

Система самодиагностики предоставляла два вида информации:

1) извещение об ошибке в том или ином модуле, печатавшееся при ее обнаружении и содержавшее: обозначение модуля; наименование выполняемой диагностической программы; время, прошедшее после предыдущей ошибки в данном модуле; сведения об ошибке;

2) извещение о состоянии ВС в целом, которое содержало время начала работы системы самодиагностики, время выдачи извещения и сведения о состоянии всех модулей.

*Анализ архитектуры микроВС Cm**

Статистическая обработка информации по применению системы самодиагностики показала, что среднее время ***v*** безотказной работы ЭМ составляет ***v = 127,7 ч***.

Главным недостатком микроВС Cm* являлось использование нераспределенного диспетчера. Отказ мини-ЭВМ приводил к невозможности реализации функций ОС и, следовательно, к отказу системы как единого аппаратурно-программного ансамбля.

Описанная модификация архитектуры микроВС Cm* могла бы улучшить количественные характеристики микроВС как единого аппаратурно-программного комплекса и, в частности, положительно сказалась бы на надежности и живучести системы

8.7. Кластерные вычислительные системы

Понятие о вычислительном кластере

Кластерные ВС - разновидность мультипроцессорных систем. Такие системы получили широкое распространение уже в 90-х годах XXв.

Термин «вычислительный кластер», по-видимому, был впервые введен DEC (Digital Equipment Corporation). По определению DEC, кластер это группа компьютеров, которые связаны между собой и функционируют как единое средство обработки информации. Из приведенного определения видно, что корпорация DEC, по сути, ввела синоним термину «вычислительная система», а не особый тип средств обработки информации. для создания кластерных ВС используются и MISD-, и SIMD-, и MIMД-архитектуры, различные функциональные структуры и конструктивные решении

Архитектурные и технико-экономические платформы кластерных ВС

Кластерные ВС или вычислительные кластеры представляют варианты сосредоточенных и распределенных систем. Различают физические и логические кластеры. Первые кластеры являются специально сконфигурированными ВС из серийных аппаратурно-программных средств ЭВМ. Логические (или виртуальные) кластеры организуются в аппаратурно-программной среде вычислительных сетей. Для такой организации используются дополнительные системные средства (как правило, программные), которые превращают вычислительную сеть в средство для реализации параллельных вычислений.

Начало XXI в. ознаменовалось созданием пространственно-распределенных мультикластерных ВС как макроколлективов рассредоточенных кластеров, взаимодействующих между собой через локальные и глобальные сети (включая всемирную сеть Internet).

Технические средства для формирования кластерных ВС

В наиболее общей трактовке кластерная ВС, или кластер, это композиция множества вычислителей, сети связей между ними и программного обеспечения, предназначенная для параллельной обработки информации (в частности, реализации параллельных алгоритмов решения сложных задач). При формировании кластерной ВС могут быть использованы как стандартные промышленные компоненты, так и специально созданные средства. Однако в кластерных ВС, как правило, превалируют массовые аппаратурно-программные средства. Последнее, по существу, является принципом конструирования кластерных ВС, обеспечивающим их высокую технико-экономическую эффективность.

Программное обеспечение и области применения кластерных ВС

Широко распространённым средством для организации межсерверного взаимодействия является библиотека MPI, поддерживающая языки C и Fortran. Она используется, например, в программе моделирования погоды MM5.

Операционная система Solaris предоставляет программное обеспечение Solaris Cluster, которое служит для обеспечения высокой доступности и безотказности серверов, работающих под управлением Solaris. Для OpenSolaris существует реализация с открытым кодом под названием OpenSolaris HA Cluster.

Среди пользователей GNU/Linux популярны несколько программ:

- distcc, MPICH и др. — специализированные средства для распараллеливания работы программ. distcc допускает параллельную компиляцию в GNU Compiler Collection.
- Linux Virtual Server, Linux-HA — узловое ПО для распределения запросов между вычислительными серверами.
- MOSIX, openMosix, Kerrighed, OpenSSI — полнофункциональные кластерные среды, встроенные в ядро, автоматически распределяющие задачи между однородными узлами. OpenSSI, openMosix и Kerrighed создают среду единой операционной системы между узлами.

8.8. Анализ мультипроцессорных вычислительных систем

В настоящее время наблюдается необыч2.йный подъем в исследованиях и опытно-конструкторских работах по мультипроцессорным ВС с усовершенствованной структурой. Такие ВС вплотную подошли к средствам, полностью основанным на модели коллектива вычислителей, т. е. к ВС с программируемой структурой.

В отличие от систем с канонической структурой мультипроцессора в ВС с распределенной памятью принцип программируемости структуры получил внедрение в наиболее завершенном виде. Системы стали обладать способностью к автоматической реконфигурации структуры в процессе решения задач, представленных в параллельной форме, они стали допускать возможность разбиения на подсистемы для целей мультипрограммирования. Вычислительные системы приобрели большую способность к статической (априорной) реконфигурации структур и состава для адаптации под область применения и условия эксплуатации. Такие ВС становятся все более распределенными и в реализации ориентированы на современную технологию микропроцессорных БИС.

Анализ архитектуры мультипроцессорных ВС позволяет сделать нижеследующие выводы.

1. Основная тенденция в области архитектуры мультипроцессорных ВС повышение степени полноты воплощения принципов модели коллектива вычислителей (параллелизма, программируемости структуры и конструктивной однородности).

2.Архитектурные возможности ЭП неуклонно наращиваются, их структура претерпела трансформацию от простейших конфигураций без памяти до элементарных машин композиции из мощных микропроцессоров, оперативной памяти и внешних запоминающих устройств (и даже устройств ввода-вывода информации).

3. Мультипроцессорные ВС, начавшие свою историю как композиции из нескольких процессоров, превратились в системы с массовым параллелизмом (с количеством процессоров порядка 10^2-10^5).

4. Современные мультипроцессорные ВС - это распределенные средства обработки информации, они имеют множество процессоров и распределенную память. Более того, в них и коммутатор (или другой ресурс), через который осуществляется взаимодействие

процессоров, может быть распределенным. Программное обеспечение таких ВС также : является распределенным.

5. Высокопроизводительные ВС рассматриваемого класса представляют собой суперсистемы: это множество мощных микропроцессоров-конвейеров или матричных процессоров или даже объединение мультипроцессоров.

ВЫЧИСЛИТЕЛЬНЫЕ СИСТЕМЫ С ПРОГРАММИРУЕМОЙ СТРУКТУРОЙ

9.1. Понятие о вычислительных системах с программируемой структурой

Определение ВС

Вычислительные системы с программируемой структурой - это распределенные средства обработки информации. В таких ВС нет единого функционально и конструктивно реализованного устройства: все компоненты (устройство управления, процессор и память) являются распределенными. Тип архитектуры ВС MIMD ; в системах заложена возможность программной перенастройки архитектуры MIMD в архитектуры MISD или SIMD

Основной функционально-структурной единицей вычислительных ресурсов в системах рассматриваемого класса является элементарная машина (ЭМ). допускается конфигурирование ВС с произвольным числом ЭМ. Следовательно, ВС с программируемой структурой относятся к масштабируемым средствам обработки информации и допускают формирование конфигураций с массовым параллелизмом.

При построении ВС с программируемой структурой доминирующими являются следующие три принципа:

1. массовый параллелизм (параллельность выполнения большого числа операций);
2. программируемость (автоматическая перестраиваемость или реконфигурируемость) структуры;
3. конструктивная однородность.

Принцип программируемости структуры требу ет, чтобы в ВС была реализована возможность хранения программного описания функциональной структуры и программной ее модификации (перенастройки) с целью достигнуть адекватности структурам и параметрам решае мых задач.

Под ВС с программируемой структурой понимается совокупность элементарных машин, функциональное взаимодействие между которыми осуществляется через программно настраиваемую сеть связи. Элементарная машина это композиция из вычислительного модуля и системного
устройства.

Вычислительный модуль (ВМ) служит как для переработки и хранения информации, так и для выполнения функций по управлению системой в целом. Системное

устройство (СУ) аппаратурная часть ЭМ, предназначенная только для обеспечения взаимодействия данной ЭМ с ближайшими соседними машинами.

Сосредоточенные и распределенные ВС

Характерной особенностью сосредоточенных ВС является компактное пространственное размещение средств обработки и хранения информации, при котором нет необходимости использовать телекоммуникационные сети (существующие абонентские или специально разработанные сети передач данных). В сосредоточенных ВС нет линий связи, вносящих существенную задержку в работу ВС, нет жестких ограничений на топологию сети связи, на возможность параллельной передачи информации между функциональными модулями (процессорами, модулями памяти, ЭМ и др.).

К распределенным ВС относят макросистемы - системы сложной конфигурации, в которых в качестве функциональных элементов выступают пространственно-рассредоточенные вычислительные средства, основанные на моделях вычислителя и коллектива вычисл: отелей, и сети связи, обеспечивающие взаимный теледоступ между средствами обработки информации.

Распределенная ВС объединение пространственно удаленных друг от друга сосредоточенных ВС, основанное на принципах:

- параллельности функционирования сосредоточенных ВС
- программируемости структуры
- гомогенности состава (программной совместимости различных сосредоточенных ВС и однотипности элементарных машин в каждой из них)

Предельным по простоте вариантом сосредоточенной ВС является ЭМ, следовательно, простейшим вариантом распределенной ВС является совокупность однотипных и регулярно соединенных ЭМ, использующая «длинные» программно-коммутируемые каналы межмашинной связи. в этом случае в состав ЭМ входят однотипные СУ.

Распределенные системы относятся к более общему классу средств обработки информации по сравнению с вычислительными сетями и качественно от них отличаются возможностью решения одной сложной задачи на распределенных вычислительных ресурсах.

Распределенные ВС в сравнении с компьютерными сетями являются более общим классом средств переработки информации.

Распределенные системы, в которых выполняется условие программной совместимости ЭМ, относятся к гомогенным ВС.

9.2. Архитектурные особенности вычислительных систем с программируемой структурой

9.2.1. Структура ВС

Требования, предъявляемые к структуре ВС

1) Простота вложения параллельного алгоритма решения сложной задачи в структуру ВС

Структура ВС должна быть адекватной достаточно широкому классу решаемых задач

2) Удобство адресации ЭМ и переноса подсистем в пределах ВС

Структура ВС должна позволять реализовать простейший механизм преобразования виртуальных адресов ЭМ в реальные (физические) адреса машин

3) Осуществимость принципа близкодействия и минимума задержек при межмашинных передачах информации в ВС

Структура должна обеспечивать минимум задержек при транзитных передачах информации

4) Масштабируемость и большемасштабность структуры ВС

Структура должна обладать способностью к наращиванию и сокращению числа вершин, при этом изменение числа ЭМ в ВС не должно приводить к коренным перекоммутациям между машинами

5) Коммутируемость структуры ВС

Структура ВС должна обладать способностью осуществлять заданное число одновременно непересекающихся взаимодействий между ЭМ

6) Живучесть структуры

Способность структуры ВС обеспечивать связность требуемого числа работоспособных ЭМ в системе при ненадежных линиях связи

7) Технологичность структуры ВС

Структура не должна предъявлять особых требований к элементной базе и к технологиям изготовления микропроцессорных БИС

Структурные характеристики ВС (диаметр, средний диаметр, вектор-функции структурной коммутируемости и живучести ВС)

Диаметр – максимальное расстояние, определенное на множестве кратчайших путей между двумя вершинами структуры ВС

$$d = \max_{i,j}\{d_{ij}\}$$

Средний диаметр

$$đ = (\mathrm{N}-1)^{-1}\sum_{l=1}^{d} l - n_l$$

d_{ij} – расстояние, т. е. минимальное число ребер, образующих путь из вершины i в вершину j

$$i,j \in \{0,1,2,\dots,N-1\}$$

n_l – число вершин, находящихся на расстоянии l от любой выделенной вершины однородного графа G

Структурная коммутируемость определяется вектор-функцей $K(G,S,S') = \{K_h(G,S,S')\}$

$$h \in \{1,2,\dots,\left\lceil\frac{N}{2}\right\rceil\}$$

$K_h(G,S,S')$ – вероятность реализации в системе при заданной структуре G и коэффициентах (S – готовности одной ЭМ и S' - коэффициент готовности в линиях связи h) есть вероятность h одновременно непересекающихся взаимодействий

Структурная живучесть оценивается вектор-функцией $L(G,S,S') = \{L_r(G,S,S')\}$

$$r \in E_2^N = \{2,3,\dots,N\}$$

$L_r(G,S,S')$ - вероятность существования подсистемы ранга r, т. е. подмножества из r работоспособных ЭМ, связность которых устанавливается через работоспособные линии связи при заданных при заданных: структуре G и коэффициентах (S – готовности одной ЭМ и S' - коэффициент готовности в линиях связи h)

Перспективные структуры ВС (D_n – графы) и Л(N, v, g - графы)

1) **Циркулянтная структура D_n-граф**

$$\{N;\ \omega_1;\ \omega_2 \dots;\ \omega_n\}$$

N – количество вершин. Вершины помечены целыми числами

$$i \dots N,$$

$$i \in \{0,1,\dots,N-1\},$$

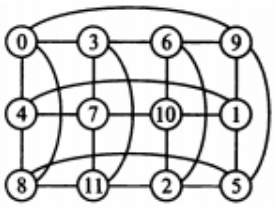
$i\pm, i \pm \omega_2, \dots, i \pm \omega_n (mod\ N)$ – множество образующих, таких, что

$0 < \omega_1 < \omega_2 < \dots < \omega_n < \frac{N+1}{2}$

$$N, \omega_1, \dots, \omega_n \text{ НОД} = 1$$

n – размерность графа

2n – степень вершины



D_2-граф: {12; 3, 4}

Целые числа i, отмечающие вершины D_n-графа, называют адресами. Адресация вершин в таких структурах называется диофантовой. В циркулянтных структурах при полном переносе какой-либо подструктуры (всех вершин подструктуры на одно и то же расстояние в одном из направлений) сохранятся все ее свойства и адресация вершин.

При диофантовой адресации элементарных машин ВС можно простыми средствами реконфигурации осуществить виртуальную адресацию вершин-машин и, следовательно:

1) создавать отказоустойчивые параллельные программы, неориентированные на физические номера машин;

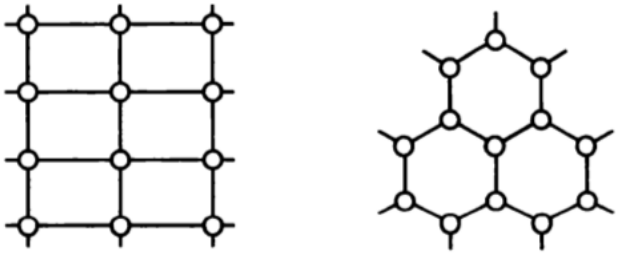
2) реализовывать мультипрограммные режимы обработки информации;

3) исключать отказавшие вершины-машины из подсистем, а значит, обеспечить живучесть ВС.

2) **Л(N,V,g)-граф** – неориентированный однородный граф с числом вершин N, степенью вершины V и значением обхвата g

g – длина кратчайшего цикла в графе

В $Л(N,V,g)$-графах каждая вершина при $V \geq 3$ входит в не менее V кратчайших простых циклов длиной g. $Л(N,V,g)$ допускают масштабирование числа машин без коренной перекоммутации существующих межмашинных связей



а *б*

Рис. 7.3. Фрагменты $Л(N, v, g)$-графов:
$а$ — $v = 4,\ g = 4;$ $б$ — $v = 3,\ g = 6$

Анализ и синтез структур ВС

Введенные показатели структурных характеристик ВС в равной степени пригодны и для анализа, и для синтеза структур ВС, т. е. требуется осуществлять расчет значений структурных характеристик ВС. Получение аналитических выражений для координат вектор-функций структурной коммутируемости и структурной живучести является сложной задачей, разрешимой лишь для частных случаев. Для их расчета используют метод статистического моделирования.

Проблема синтеза структур может быть сформулирована с ориентацией на любой из структурных показателей. Можно дать следующую постановку: найти структуру G^*, которая бы обеспечивала максимум координаты вектор-функции структурной живучести ($\max_G L_r(G,S,S') = L_r(G^*)$), т. е. G^*, для которой выполняется это условие, является оптимальной при заданных $\mathrm{N}, \mathrm{r}, \mathrm{V}, S, S'$

Гипотеза 1

Структура G^*, при которой достигается $L_N(G^*)$, обеспечивает и $L_r(G^*)$ - максимум живучести подсистемы ранга r, r<N

Гипотеза 2

Структура с минимальным средним диаметром относится к G^*, т. е. обладает максимальной структурной живучестью

9.2.2. Режимы функционирования ВС и способы обработки информации

Основные режимы работы ВС с программируемой структурой:

- решение одной сложной задачи;
- обработка набора задач;
- обслуживание потока задач.

Первый режим монопрограммный, для решения задачи используются все ресурсы ВС. Задача представляется в виде параллельной программы, число ветвей в которой либо фиксировано, либо допускает варьирование в заданном диапазоне. В качестве единицы ресурса выступает элементарная машина ВС.

Проблемы при решении с точки зрения архитектуры

- Распараллелить программу
- Организовать отказоустойчивое выполнение

- Эффективно вложить ветви ПП в структуру системы, так чтобы взаимодействие между ними происходило как можно быстрее.
- Не все задачи можно распараллелить, то есть необходимо решить проблему простоев системы.

Второй и третий режимы функционирования ВС относятся к мультипрограммным. При работе ВС в этих режимах одновременно решается несколько задач, ресурсы системы делятся между несколькими задачами.

При организации функционирования ВС в случае набора задач учитывается не только количество задач, но их параметры: число ветвей в программе (точнее, число машин, на которых он а будет выполняться), время решения или вероятностный закон распределения времени решения и др. Алгоритмы организации функционирования ВС задают распределение задач по машинам и последовательность выполнения задач на каждой машине.

Третий режим обслуживание потока задач на ВС принципиально отличается от обработки задач набора: задачи поступают в случайные моменты времени, их параметры случайны.

Технология решения произвольной задачи на ВС

- выбор способа обработки данных
- разработку параллельного алгоритма
- запись (параллельного) алгоритма решения задачи на языке (высокого уровня)
- получение объектной программы решения задачи на системе

Различают распределенный, матричный, обобщенный матричный и конвейерный способы обработки информации

При распределенной обработке параллельные программы и данные рассредотачиваются по элементарным машинам ВС. Допустимо построение адаптирующихся параллельных программ, число ветвей в которых в процессе их реализации соответствует числу (работоспособных) ЭМ в системе.

В случае матричной обработки данных программа вычислений содержится в одной (управляющей) ЭМ, а данные однородно распределяются по всем машинам ВС (или подсистемы). Процесс решения задачи состоит из чередующихся процедур: рассылки команд из управляющей ЭМ остальным машинам и исполнения этих команд всеми машинами, но каждой над своими операндами.

Обобщенный матричный способ обработки информации. При этом способе программа не целиком помещается в одной ЭМ, а предварительно сегментируется и затем посегментно размещается в памяти машин. Последовательность сегментов, составляющих программу, может быть размещена в памяти машин, например так, что номер распределенного в машину сегмента будет равен ее номеру.

При конвейерном способе обработки данных структура ВС предварительно настраивается так, что машины образуют конвейер (или «линейку», или «кольцо»). Затем осуществляются сегментирование программы и размещение в машинах ВС последовательности полученных сегментов в соответствии со структурой конвейера. Размещен сие данных может быть сосредоточенным (например, на внешней памяти одной ЭМ) или распределенным (по памятям всех машин конвейера). В процессе решения задачи данные проходят через последовательность машин, составляющих конвейер.

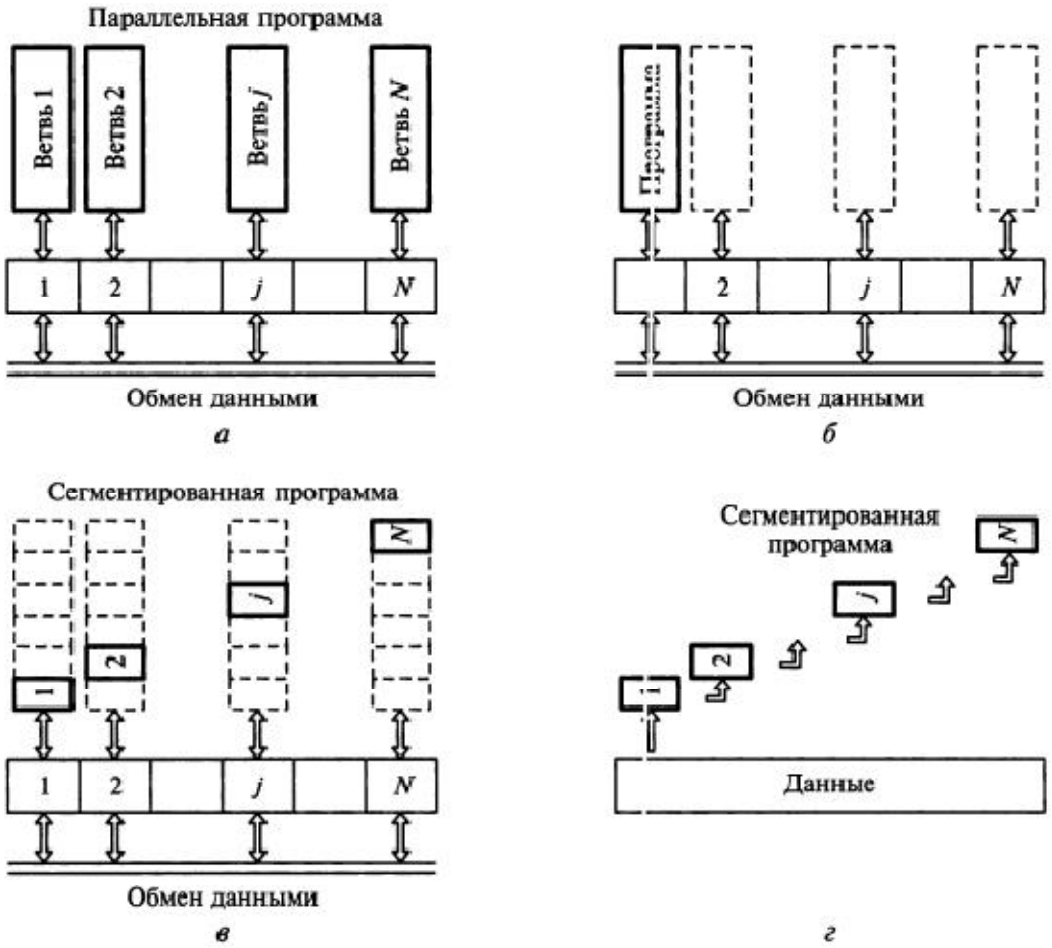


Рис. 7.7. Способы обработки информации:

а — распределенный; *б* — матричный; *в* — обобщенный матричный; *г* — конвейерный;
⇨ — направление потоков данных

9.2.3. Архитектурные аспекты при создании операционных систем ВС

В основе организации параллельных вычислений в ВС лежит представление вычислении в виде совокупности совместно протекающих асинхронных взаимодействующих процессов.

В общем случае для ВС характерен мультипрограммный режим работы даже при решении задач, представленных в параллельной форме. Мультипрограммирование является следствием разделения ресурсов ВС между:

- процессами, относящимися к различным параллельным программам пользователей;
- процессами пользовательских программ и процессами операционной системы;
- отдельными процессами, относящимися к одной программе решения задачи.

Управление работой ВС в мультипрограммном режиме возлагается на ядро (резидентную часть) ОС и непосредственно осуществляется с помощью операции управления процессами.

Базовый набор средств, обеспечивающий управление процессами каждой ЭМ ВС, составляют средства для порождения и уничтожения процессов и для реализации взаимодействий между ними. На основе базового набора строятся все процессы ОС элементарной машины ВС : управление распределением ресурсов, связь с внешними устройствами и с пользователями, связь с ОС других ЭМ.

Временное и пространственное распределение аппаратурно-программных ресурсов системы (процессоров, оперативной памяти, внешних устройств, системных устройств, линий связи, программ, данных) между совместно протекающими процессами обеспечивается благодаря полноте реализации в ВС трех канонических принципов модели коллектива вычислителей. Так, программируемость структуры позволяет уже на этапе программирования сложной задачи не только указывать последовательность реализации процессов (подчиненность процессов по данным и по управлению), но и задавать размещение этих процессов в пространстве ресурсов ВС.

Программируемость структуры ВС достигается аппаратурно-программными средствами, среди которых уже известные системные устройства и специальные программные блоки ОС.

Все отмеченные архитектурные возможности достигаются в ВС с программируемой структурой с помощью специальных средств, которые в комплексе с традиционными средствами организации функционирования и составляют ОС.

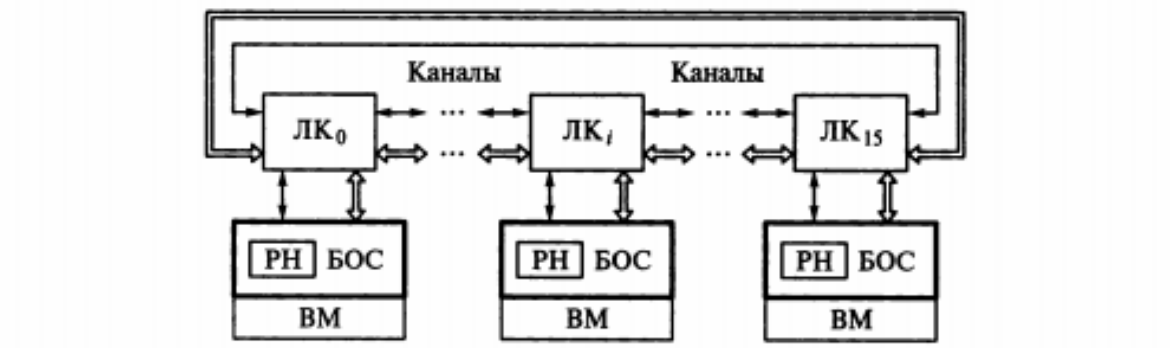
В общем случае ОС имеет иерархическую структуру.

9.3. Вычислительная система "Минск-222"

Разработана и построена в ИМ СО АН СССР совместно с конструкторским бюро завода имени Орджоникидзе в 1965-66 гг.

Архитектура MIMD. Параллелизм, однородность и программируемость структуры. Одномерная кольцевая топология. Масштабируемость от 1 до 16 ЭМ. В качестве ЭМ использовались промышленные ЭВМ второго поколения.

9.3.1. Функциональная структура ВС "Минск-222"



ЛК — локальный коммутатор; БОС — блок операций системы; РН — регистр настройки; ВМ — вычислительный модуль; ⇨ — рабочий канал; → — управляющий канал

Элементарная машина

Элементарная машина состояла из вычислительного модуля и системного устройства (СУ). В качестве ВМ были использованы серийные ЭВМ «Минск-2» или «Минск-22». Указанные ЭВМ имели одну и ту же архитектуру и были не только совместимы, а, по сути, являлись конфигурациями одной и той же двухадресной машины.

Системное устройство

В состав СУ входили локальный коммутатор (ЛК) каналов связи и блок операций системы (БОС).

В качестве ЛК используются вентили, которые открываются или закрываются канал связи идущий к машине справа.

Содержимое РН определяет соединительные функции коммутатора и степень участия ЭМ при системном взаимодействии. РН состоит из 3-х разрядов TR, TQ, TΩ

- TR – разбивает систему на подсистемы («1» - есть связь, «0» - нет связи) | TR = R0…R15
- TQ и TΩ - конкретизирует степень участия машины в выполнении системных команд.
- TQ – определяет участие машины в выработке обобщенного признака Ωк; к = 1,2,3; Ωк = &i€Eωki

ω1i ; k = 1 – если знак последнего результата < 0
ω2i ; k = 2 – если число было переполнено
ω3i ; k = 3 – если результат равен 0
Ωк, к = 1 – если во всех предыдущих машинах результат < 0
Е – подмножество номеров машин управляющих ходом вычислений.

ВС отличалась от ЭМ тем, что содержала в себе локальный коммутатор и БОС
Системное устройство было реализовано на 80 стандартных элементах и составляло менее 1, 5 % объема оборудования АЛУ и устройства управления ЭВМ «Минск-22»

9.3.2. Системные команды ВС "Минск-222"

Система команд ЭВМ была расширена командами настройки, т. Е. программируемой структурой, командами обмена информации, командами обобщенного условного и безусловного переходов.

Команды представлялись 37 разрядными двоичными числами

+- 0	1 КОП 6	7 Θ 12	13 A_1 24	25 A_2 36

КОП – код операции
Θ – шести разрядное поле, 2 разряда которого определяли номер блока памяти, оставшиеся 4 – номер блока ячейки.
A_1 и A_2 – адреса

Команды настройки

Общий вид команды настройки

1	01	00	A_1	A_2

Команда настройки включала три модификации в зависимости от значения 29 и 34 бита.

29 и 34 = 0, H_0 – эта команда изменяет содержимое регистра настройки, только своей ЭМ. При этом новое содержимое Регистров R, Q и Ω находятся в 31, 32 и 33 разрядах.

29 = 1, 34 = 0, H_1 – изменяет содержимое РН тех ЭМ, которые указаны с 13 по 28 разряд. Номер машины = номер разряда – 13. При этом содержимое собственного регистра настройки не изменяется.

29 = 0, 34 = 1, H_2 – изменяет содержимое РН только для своей ЭМ и изменяет содержимое регистра округления и признака ω_{3i} , 35 – регистр округления, 36 - ω_{3i}

Команды обмена

Команда обмена

Передача

1	36	00	*l*	α

Прием

1	57	00	h	β

l – количество передаваемых слов

α – указатель начала буфера для передачи

h – количество принимаемых слов

β – указатель начала буфера для записи

Команда приема блокирующая (не будет завершена, пока не принято h слов).

Команды обобщенного безусловного перехода

Обобщенный безусловный переход

1	02	$A_7$0	00	A_2

Содержимое бита 7-го разряда определяло тип операции. Выделим две модификации: $ОБП_0$ и $ОБП_1$. При выполнении $ОБП_0$ следующая инструкция выполнялась на всех машинах после окончания текущей не вытесняя. $ОБП_1$ вытесняла. $ОБП_0$ использовалась для синхронизации выполнения команд. С помощью команд $ОБП_0$ и $ОБП_1$ из любой машины в любую может быть передана команда для включения.

Команды обобщенного условного перехода

Команда условного перехода

1	65	00	A_1	A_2

Команда условного перехода выполняется только в тех машинах, в которых $\Omega = 1$. Выделяют несколько операций условного перехода i = 0,1,2,3 вид операции вид операции определяют 13,14,15 бит.

Если i = 0, то осуществляется переход к следующей команде. В случае i = 1,2,3 происходит передача управления либо по адресу в A_2, либо к следующей команде

9.3.3. Программное обеспечение ВС "Минск-222"

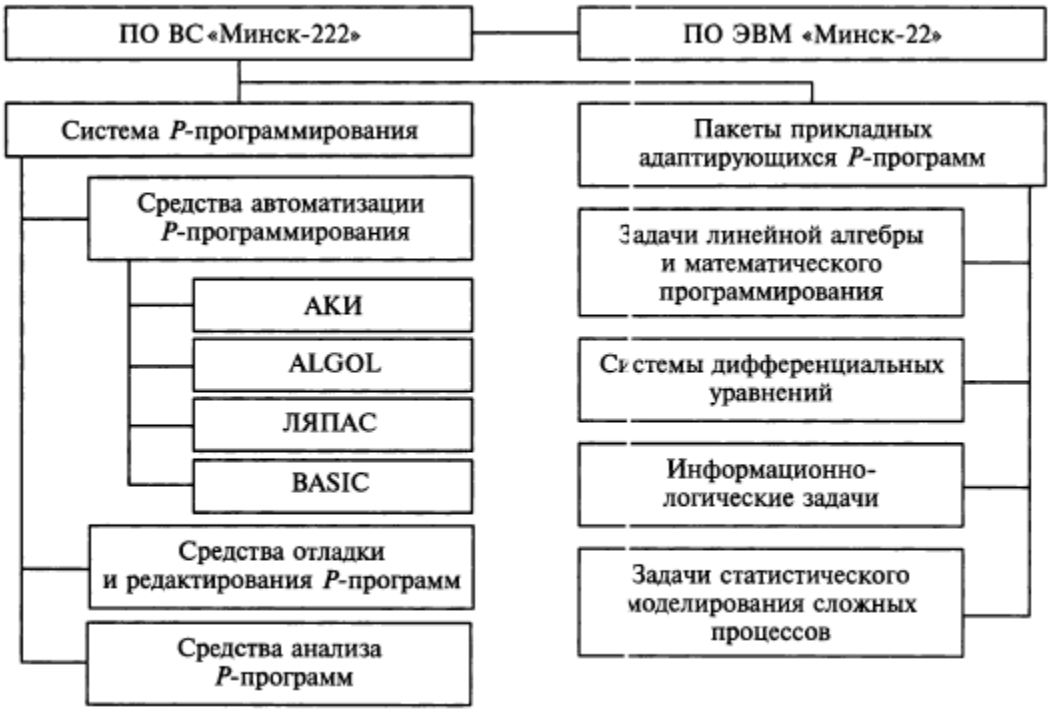


Рис. 7.9. Программное обеспечение ВС «Минск-222»

Система Р – программирования

Это расширенное ПО МИНСК-22 состояло из системы параллельного программирования и пакетов прикладных адаптируемых программ. Система параллельного программирования (ПП) включала: Средства автоматизации ПП, средства отладки, редактирования и анализа ПП.

Средства автоматизации ПП – языки и трансляторы. В качестве языков использовались: Автокод АКИ, ЛЯПАС, ALGOL, Basic.

Разработка транслятора для расширенных языков программирования свелась к разработке системных блоков. Любой системный блок представлял совокупность программ для реализации операций настройки, обмена, ОБП, ОУП, которые были включены в библиотеку трансляторов.

Средство отладки и редактирования – совокупность 4-х стандартных программ.

1. Преобразовывала отлаживаемую ПП в последовательную для выявления ошибок не связанных с использованием системных команд.
2. Служила для моделирования на одной машине выполнения программы из 2-х ветвей.
3. Позволяла вывести на печать заданное количество раз содержимое интересующих областей памяти.
4. Служила для корректировки ПП.

Средство анализа состояла из 3-х программ

1. Служила для анализа распределения памяти между блоками исследуемых программ.
2. Для измерения времени простоев машин вычислительной системы
3. Применялась для измерения времени работы участков ПП.

Пакеты прикладных адаптирующихся Р – программ

Пакеты прикладных адаптирующихся ПП ориентированы на решения задач повышенной сложности. Параметры таких задач не позволяли решать их на ЭВМ. Под адаптирующиеся понимаются программы, которые могут настраиваться на число элементарных ВС как на параметр.

9.3.4. Области применения и эффективность ВС "Минск-222"

Система «Минск-222», имевшая программируемую структуру, позволяла решать задачи, представленные как последовательными, так и параллельными программами с различными количествами ветвей.

- Решение систем линейных алгебраических уравнений (методом простой итерации, методом Жордана, методом Зйделя, методом Самарского)
- обращение матриц методом пополнения;
- отыскание коэффициентов характеристического полинома;
- умножение матриц
- решение систем нелинейных алгебраических уравнений
- численное интегрирование
- численное дифференцирование и т.д.

В параллельных программах решенных задач были использованы известные схемы обмена информацией между ветвями: трансляционная Т («одна всем»), трансляционно-циклическая ТЦ («каждая всем»), конвейерно-параллельная КП («каждая своим соседям»), коллекторная К («все одной») и дифференцированная Д («любая любой»). Причем первые три схемы обмена составляют 90% используемых.

Было установлено, что для ВС «Минск-222» доля затрат времени на системные взаимодействия (включая синхронизацию) составляет, как правило, несколько процентов, что является следствием применения методики крупноблочного распараллеливания задач.

Парадокс параллелизма был впервые обнаружен при работе на ВС <Минск-222».

Экспертные оценки показали, что сложность программирования для ВС «Минск-222» (по сравнению со сложностью программирования для одной ЭВМ «Минск-22 ») возрастает на 10...20 %, а при развитой библиотеке стандартных параллельных программ на 5.. .10 % .

9.4. Вычислительная система МИНИМАКС

Вычислительные системы, которые формировались из аппаратурно-программнъх средств мини-ЭВМ, относились к группе мини - ВС.

Работы по созданию ВС из минимашин достаточно интенсивно велись в США. Однако общей концепции построения таких систем американские специалисты не выработали. Анализ проектов показывает, что использовались в основном три способа организации ВС:

- системы с общей памятью;
- ВС с общей шиной (или системой шин), к которой подключались процессоры, запоминающие и другие устройства;

- системы, в которых машины взаимодействовали через общую группу устройств ввода-вывода информации. Как правило, системы не имели программируемой структуры и обладали ограниченными возможностями к наращиванию.

Технический проект МИНИМАКС разработан в 1974 г., а опытно-промышленный образец системы был изготовлен и отработан в 1975 г.

Архитектура системы МИНИМАКС:

- распределенность средств управления, обработки и памяти;
- параллелизм, однородность, модульность
- программируемость структуры;
- двумерная (циркулянтная) топология;
- масштабируемость;
- живучесть;
- максимальное использование промышленных средств мини-ЭВМ.

9.4.1. Функциональная структура мини-ВС МИНИМАКС.

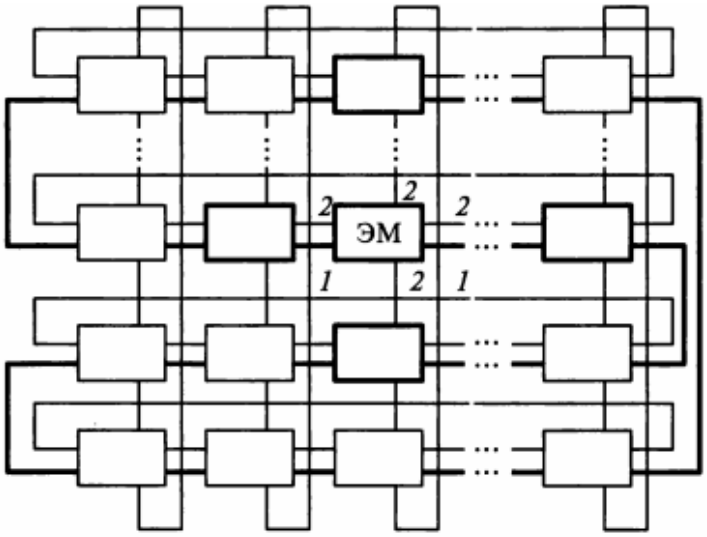


Рис. 7.10. Функциональная структура мини-ВС МИНИМАКС:

ЭМ — элементарная машина; *1* — одномерные управляющие каналы; *2* — двумерные рабочие каналы

Функциональная структура мини-ВС МИНИМАКС композиция из произвольного количества элементарных машин и программно настраиваемой сети связей между ними. Мини-ВС МИНИМАКС обладала свойством масштабируемости: в ней количество ЭМ не было фиксировано и определялось сферой применения.

Взаимодействия между ЭМ в системе МИНИМАКС осуществлялись через сеть связей (рис. 7.10), которая формировалась из одномерных 1 и двумерных 2 полудуплексных каналов. Одномерные каналы связи 1 были управляющими; они служили для программирован я соединений между ЭМ по каналам связи 2, а также для передачи между ЭМ управляющей информации, регламентирующей использование общих ресурсов (внешних устройств, сервисных программ, файлов и т. п.). Двумерные каналы связи 2 являлись рабочими; они применялись для следующих целей:

- реализации основных межмашинных взаимодействий

- пересылки массивов данных между памятями передающей ЭМ и одной или нескольких принимающих ЭМ
- передачи адресов из одной ЭМ в другую и обмена логическими переменными между машинами.

Системные взаимодействия, осуществлявшиеся в мини-ВС, были следующих типов:

- программное изменение структуры мини-ВС и степени участия ЭМ в системных взаимодействиях (настройка)
- передача информации из оперативного запоминающего устройства одной ЭМ в памяти других (обмен)
- выполнение ОБП
- синхронизация работы ЭМ
- реализация ОУП по значению признака Ω

Межмашинные взаимодействия при функционировании мини-ВС реализовывались с помощью специальных подпрограмм - системных драйверов, которые, в свою очередь, использовали специальные команды (занесение кода на регистр настройки, считывание его содержимого, занесение информации в СУ о начальном адресе передаваемого массива данных и т. п.). СУ – системное устройство.

Структура системы МИНИМАКС, приведенная на рис. 7.10, была удобна для реализации параллельных программ сложных задач. Однако в определенных сферах применения мини-ВС МИНИМАКС требовались оптимальные структуры. Поэтому при компоновке мини-ВС, состоявшей из N ЭМ, использовались для отождествления полюсов связей 1 (см. рис. 7.10) одномерные структуры, а для отождествления полюсов связей 2 оптимальные двумерные структуры, точнее, D_2-графы.

В пределах мини-ВС МИНИМАКС допускалось формирование произвольного числа подсистем из любого количества ЭМ. Подсистему составляли взаимодействовавшие друг с другом ЭМ вместе с машинами, которые использовались в качестве транзитных пунктов передачи информации. Каждая ЭМ могла входить только в одну из подсистем (ПС2), образованных по связям 2. Вместе с тем она могла входить и в одну из подсистем (ПС 1), образованных по связям 1. Машина, принадлежавшая подсистемам ПС1 и ПС2, не могла одновременно участвовать в нескольких взаимодействиях. Подсистемы ПС2 могли сохраняться в течение нескольких следующих друг за другом взаимодействий. Подсистемы ПС1 образовывались только на время одного взаимодействия и разрушались после его выполнения.

9.4.2. Элементарная машина и системное устройство мини-ВС МИНИМАКС.

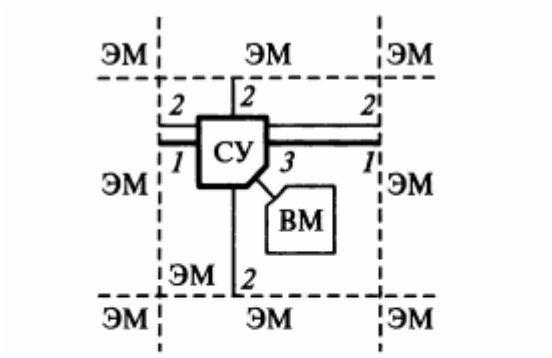


Рис. 7.12. Функциональная структура ЭМ мини-ВС МИНИМАКС:

ВМ — вычислительный модуль; ЭМ — элементарная машина; СУ — системное устройство; *1–3* — связи

Вычислительная система МИНИМАКС компоновалась из ЭМ-многополюсников (рис. 7.12). Каждая ЭМ это композиция из ВМ и СУ. Структура ЭМ данной мини-ВС не была жестко заданной и определялась областью применения. Состав каждой ЭМ допускал варьирование; компоновка ЭМ проводилась по правилам, которые были приняты для агрегатных средств ВТ на микроэлектронной основе (АСВТ-М) – (Агрегатная система средств вычислительной техники) или для средств системы малы ЭВМ (СМ ЭВМ).

В качестве ВМ могли быть использованы любые конфигурации мини-ЭВМ на базе процессоров М-6000, М-7000, СМ-1 П. Архитектура системы МИНИМАКС была рассчитана также на применение мини-ЭВМ моделей HP 2114-2116 семейства Hewlett Packard.

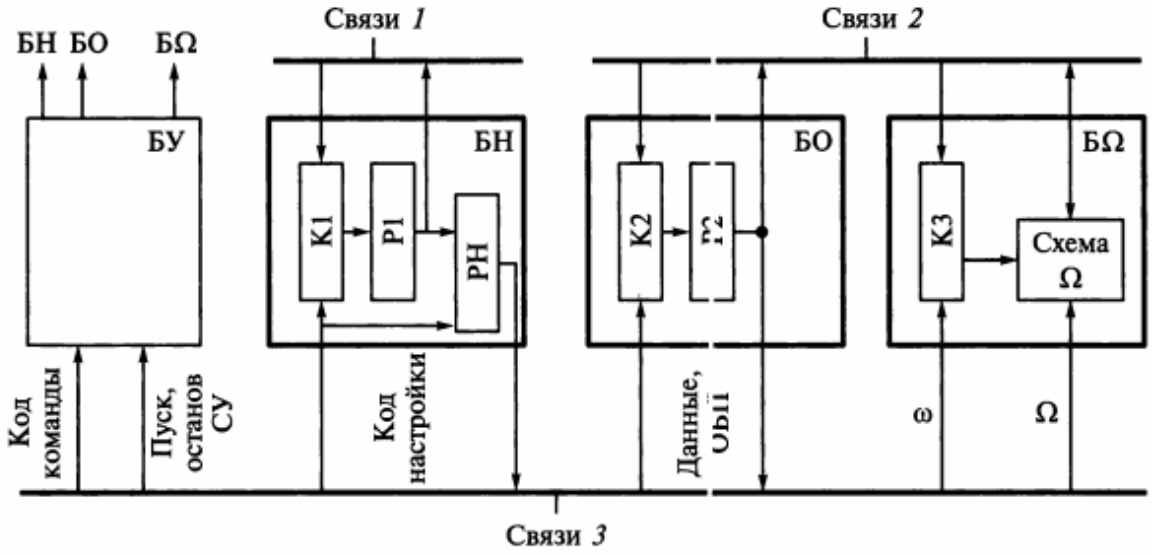


Рис. 7.13. Функциональная структура системного устройства мини-ВС МИНИМАКС:

БН — блок настройки; БО — блок обмена информацией; БУ — блок управления; БΩ — блок выработки обобщенного признака; $К_1$, $К_2$ — коммутаторы; $РН_1$, $РН_2$ — регистры настройки

Системное устройство было спроектировано как автономное устройство АСВТ-М. Оно подключалось к ВМ через связи 3 (рис. 7.13). При выборе способа реализации связей 3 учитывались принципы построения АСВТ-М и следующие отсюда ограничения:

- целесообразность построения системного устройства в виде отдельного модуля;
- недопустимость изменении в схемах и конструкции процессоров АСВТ-М

В мини-ВС МИНИМАКС связь ВМ с СУ реализовывалась двумя способами :

- через программный канал;
- через канал прямого доступа к памяти.

При этих способах использовался стандартный интерфейс ввода-вывода АСВТ-М.

В состав СУ входили: блок управления БУ, блок настройки БН, блок обмена информацией БО, блок выработки обобщенного признака Ω БΩ.

Блок управления служил для реализации алгоритма работы системного устройства в соответствии с кодами системных операций. Управляющие сигналы данного блока координировали работу остальных БН, БО, БΩ.

Блок настройки. Настройка ЭМ заключалась в изменении кода в ее РН. Содержимое 3 разрядного РН указывало, выполняет ли данная ЭМ взаимодействия по связям 2 и задает ли направление приема информации для взаимодействий 2 и 3 (два разряда для кода соединительной функции). Настройка могла быть выполнена из собственного ВМ через связи 3 (настройка машины) или из любой другой ЭМ через связи 1 (настройка системы).

Блок обмена. Этот блок предназначался для выполнения передач информации между оперативными памятями машин, выделенных при настройке, и для пересылок адресов передачи управления при операции обобщенного безусловного перехода

Блок Ω. Блок Ω служил для синхронизации машин и осуществления обобщенного условного перехода в мини-ВС.

Контроль. Информация, передаваемая по связям 1-3, контролировалась по паритету. Следовательно, вычисления по модулю 2 контрольных сумм кодов осуществлялись до их передачи и после приема. Если эти контрольные суммы не совпадали (одна была четной, а другая нечетной), то имела место ошибка. Она вызывала выработке у сигнала прерывания в соответствующем ВМ. Реакция мини-ВС на возникновение ошибки определялась содержимым регистра настройки и источником ошибки (связи 1,2 и т. д.).

9.4.3. Системные команды мини-ВС МИНИМАКС

Межмашинные взаимодействия при функционировании мини-ВС реализовывались при помощи системных драйверов и команд. Системный драйвер специальная подпрограмма, которая выполнялась в ВМ и осуществляла:

- формирование кода системной команды в соответствии с требованиями к ее формату;
- выдачу в СУ кода команды и сигналов инициирования ее выполнения;
- контроль за ходом выполнения системной команды (включая реакцию на сигналы системного устройства).

Кратко опишем системные команды мини-ВС МИНИМАКС.

Настройка. Подразделялась на настройку машины и настройку системы. Настройка машины проводилась с помощью одной из двух команд. Первая осуществляла занесение кода, заданного в команде, на регистр настройки, вторая перепись содержимого

регистра Р1 на регистр настройки. Настройка системы выполнялась при помощи четырех команд, определявших направление настройки (вправо или влево от настраивающей ЭМ) и ее характер (занесение кода настройки на регистр настройки или Р1).

Обмен. осуществлялся при помощи команд передачи и приема. Начальный адрес участка памяти, используемого для хранения передаваемого (или принимаемого) массива, и объем массива задавался путем выполнения соответствующих драйверов в ЭМ. Машина могла перейти к выполнению очередной системной команды только после завершения передачи (приема) заданного количества слов.

Обобщенный безусловный переход использовался для передачи управления в машинах по адресу, поступавшему из управляющей машины. Для реализации обобщенного безусловного перехода использовались команды передачи адреса и приема адреса. Первая команда выполнялась в управляющей машине, вторая в управляемой ЭМ.

Синхронизация работы машин осуществлялась с помощью специальной команды. Машины, выполнявшие эту команду, вырабатывали значения индивидуальных признаков $\omega = 1$. Синхронизация достигалась, когда все синхронизируемые машины вырабатывали значения обобщенного признака $\Omega=1$.

Обобщенный условный переход реализовывался по специальной команде. Эту команду обязаны были выполнить все машины, в которых требовалось осуществить условную передачу управления по значению обобщенного признака Ω (конъюнкции индивидуальных признаков ω). Перед выполнением данной команды требовалась синхронизация работы машин. При реализации обмена, синхронизации, обобщенных условного и безусловного переходов после окончания основной команды обязательно выполнялась и вспомогательная. Функции последней команды, использовавшейся для завершения любых взаимодействий по связям 2, заключались в следующем:

- передаче информации о коллизиях в ВМ;
- приведении в исходное состояние некоторых узлов СУ.

9.4.4. Программное обеспечение мини-ВС МИНИМАКС

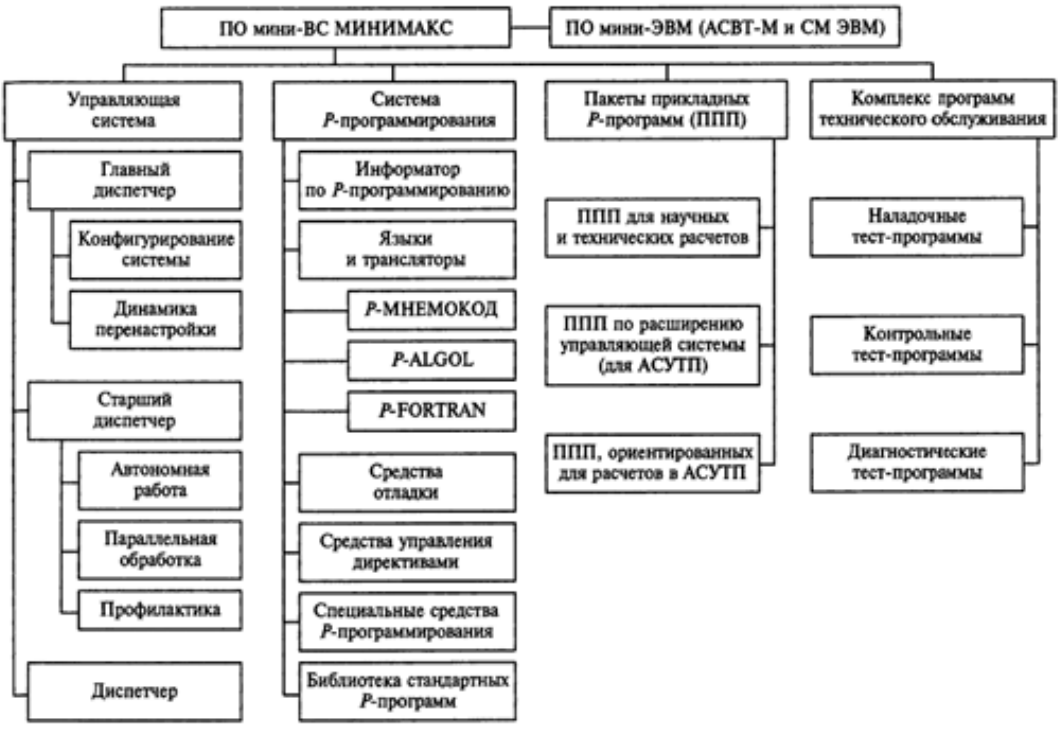


Рис. 7.14. Программное обеспечение мини-ВС МИНИМАКС

Управляющая система

Управляющая система обеспечивала связь мини-ВС с пользователями и внешними объектами, осуществляла рациональное использование ресурсов системы с учетом неисправных компонентов и входной мультипрограммной ситуации. Управляющая система имела иерархическую структуру и содержала блоки: главный диспетчер, старший диспетчер и диспетчер.

Главный диспетчер осуществлял программирование структуры и определял динамику ее перенастройки в процессе функционирования мини-ВС МИНИМАКС. Он выполнял разбиение системы на подсистемы с требуемым количеством машин и внешних устройств, формировал топологию и определял режимы работы образуемых подсистем.

Каждая подсистема могла функционировать в одном из четырех режимов: автономной работы, параллельной обработки, профилактики, управления. В режиме автономной работы осуществлялась непосредственная связь пользователя с одной машиной. В режиме параллельной обработки реализовывались Р-программы и обеспечивалось продолжение счета при выходе машин из строя. В режимах профилактики и управления выполнялись функции соответственно тестирования и организации работы системы.

Старший диспетчер организовывал функционирование подсистем в режимах автономной работы, параллельной обработки и профилактики. Он взаимодействовал с главным диспетчером.

Функции главного и старшего диспетчеров выполнялись подсистемой, работавшей в режиме управления. Диспетчеры имели столько идентичных параллельных ветвей,

сколько было машин в подсистеме. Каждая ветвь обслуживала выделенную ей сферу влияния. Основные взаимодействия ветвей были связаны:

• с дублированием некоторой информации;

• с обеспечением доступа к информации д, энной сферы по запросам из других сфер влияния;

• с перераспределением сфер влияния в связи с изменением мульти-программной ситуации.

У старшего диспетчера имелась еще одна возможность. Он хранился в той же подсистеме, что и главный диспетчер, но его функции реализовывались в других подсистемах.

Диспетчер управлял работой элементарных машин. Он участвовал в реализации системных взаимодействий (настройки, обмена, синхронизации, обобщенных условного и безусловного переходов), анализировал коллизии и аварийные ситуации, распознавал директивы пользователя, управлял индивидуальными внешними устройствами. Функции диспетчера реализовывались каждой ЭМ.

Система Р – программирования

Система Р-программирования обеспечивала достаточно широкие возможности при производстве программ для мини -ВС МИНИМАКС и включала в себя шесть нижеследующих компонентов:

1. **Информатор по параллельному программированию** являлся средством обучения пользователей способам представления вычислительных процессов в виде совокупностей ветвей, взаимодействующих между собой с помощью системных операторов.
2. **Языки Р-МНЕМОКОД , Р-ALGOL , Р-FORTRAN**, полученные путем расширения соответствующих языков операторами системных взаимодействий, применялись для записи параллельных алгоритмов
3. **Средства отладки Р-программ** служили для анализа качества программ и выявления ошибок при взаимодействии Р-ветвей (путем моделирования параллельного процесса на одной ЭМ).
4. **Средства управления заданиями** (директивами) позволяли пользователю запускать и снимать Р-программы, создавать и уничтожать Р-файлы, задавать график работы подсистем, переводить машины в режим тестирования и т. д.
5. **Средства специальной организации параллельных программ** включали:

а) сегментатор, дававший экономию оперативной памяти при хране-нии программы (он производил собственно сегментирование Р-ветви и распределение сегментов по памятям машин и организовывал последовательность реализации сегментов; следовательно, он применялся при обобщенной матричной обработке);

б) блок отказоустойчивых вычислений, обеспечивавший продолжение счета при сбоях ЭМ и выходе их из строя.

Пакеты прикладных Р – программ

Пакеты прикладных Р-программ (ППП) были ориентированы на определенные области применения мини -ВС. Среди пакетов имелись:

- пакет Р-программ общего назначения для решения научных, эко-номических и технических задач;
- пакет программ для расширения управляющей системы функциями, реализуемыми в автоматизированных системах управления технологическими процессами (АСУТП);
- пакет Р-программ, ориентированный на применение в АСУТП (например, расчет сложных поверхностей, определение технологии обработки деталей и т. д.).

Комплекс программ технического обслуживания

Комплекс программ технического обслуживания мини-ВС МИНИМАКС включал наладочные, контрольные и диагностические тест-программы.

9.4.5. Области применения мини-ВС МИНИМАКС

Ориентация агрегатных средств АСВТ-М и системы малых машин СМ ЭВМ определила области применения мини -ВС МИНКМАКС. По скольку системы МИНИМАКС имели программируемую структуру и могли состоять из произвольного количества машин, им были доступны задачи со значительным объемом вычислений.

Вычислительные системы МИНИМАКС могли применяться:

- В химической, нефтехимической, нефтеперерабатывающей, металлургической, металлообрабатывающей, приборостроительной, радиотехнической, электронной и других отраслях промышлености;
- на тепловых и атомных электростанциях и в системах энергоснабжения;
- в научно-исследовательских учреждениях, занимающихся исследованием проблем физики, гидроаэродинамики, химии, биологии, медицины и др.

Системы МИНИМАКС использовались:

- для решения научных, экономических и технических задач;
- для сбора и первичной переработки информации в сложных иерархических системах;
- для управления научными экспериментам ;
- для исследования технологических процессов и расчета технико- экономических показателей;
- для управления технологическими процессами;
- для контроля качества промышленной продукции (полупроводниковых приборов, электронных схем и др.);
- в качестве центра коммутации сообщений .

Мини-ВС МИНИМАКС могли функционировать автономно, в качестве вспомогательных подсистем мощных сосредоточенных ВС, в составе распределенных ВС или сетей.

9.5. Вычислительная система СУММА

Система СУММА была разработана Институтом математики СО АН СССР (Отделом вычислительных систем) совместно с Производственным объединением «Кварц» Министерства электронной промышленности СССР (г. Калининград).

Техническое проектирование мини-ВС было выполнено в 1975 г., опытно-промышленный образец был изготовлен и отработан в 1976 г.
Архитектура системы СУММА.

- MIMД-архитектура
- распределенность средств управления, обработки и памяти
- параллелизм, однородность, модульность
- программируемость структуры
- масштабируемость
- живучесть
- единый канал для управляющей и рабочей информации
- аппаратурно-программная реализация системных взаимодействии.

9.5.1. Функциональная структура мини-ВС СУММА

Мини-ВС СУММА формировалась из ЭМ-трехполюсников, количество которых не было фиксировано. Система характеризовалась большой архитектурной гибкостью.

Принципиальные ограничения на структуру мини-ВС (количество ЭМ и порядок их соединения) не накладывались, однако при любой структуре каждая ЭМ могла взаимодействовать не более чем с тремя соседними машинами с помощью полудуплексных каналов.

В системах управления перепрограммирование структуры мини-ВС требовалось выполнять редко, и время обмена управляющей информацией в общем времени работы машин системы составляло незначительную часть. Был единый канал для обмена управляющей (настроечной) информацией и данными между ЭМ мини-ВС.

Из-за использования для всех взаимодействий одних и тех же связей перепрограммирование структуры мини-ВС можно было осуществлять только в границах сформированных подсистем.

Для формирования мини-ВС СУММА использовались оптимальные L(N, 3, g)-графы.

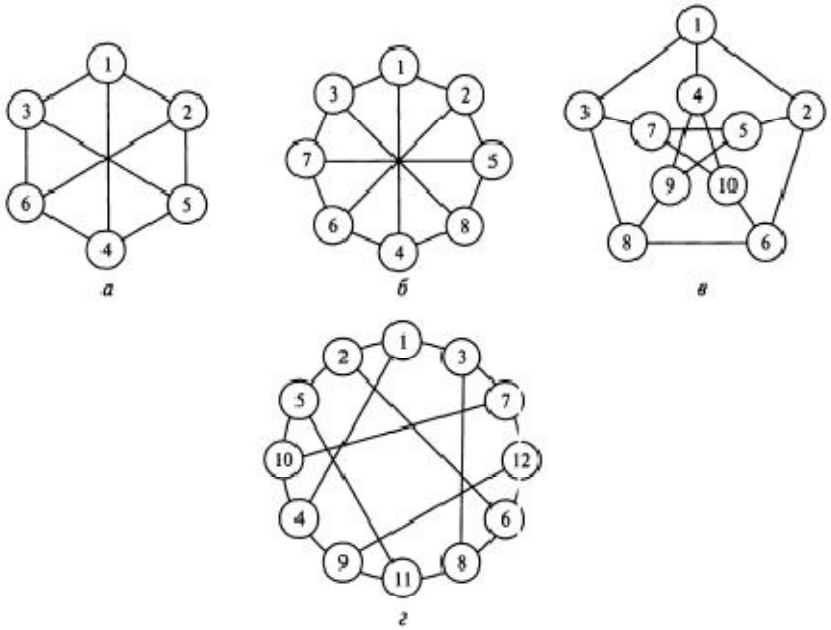


Рис. 7.15. Оптимальные структуры мини-ВС СУММА:
а — *L*(**6**, 3, 4); *б* — *L*(8, 3, 4); *в* — *L*(10, **3**, 5 ; *г* — *L*(12, 3, 5)

9.5.2. Элементарная машина и системное устройство мини-ВС СУММА

Элементарная машина системы СУММА формировалась как «трехполюсник», или, точнее, композиция из ВМ и СУ, рассчитанного на три межмашинные связи

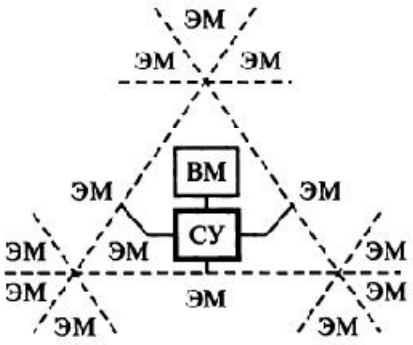


Рис. 7.16. Функциональная структура ЭМ мини-ВС СУММА:
ЭМ — элементарная машина; ВМ — вычислительный модуль; СУ — системное устройство

Вычислительный модуль предназначался для выполнения всех операций, связанных с переработкой информации, в частности для инициирования реализации системных операций. Системное устройство использовалось для реализации системных взаимодействий машин, в частности для программирования структуры мини - ВС.

В качестве ВМ использовали произвольные конфигурации мини-ЭВМ «Электроника-100 И».

Минимальная конфигурация ЭВМ «Электроника-100 И» включала процессор, ферритовую оперативную память и средства ввода-вывода информации. Состав периферийного оборудования, а также тип(магнитные ленты, диски) и объем внешней памяти определялись конкретным применением мини-машины.

Обмен информацией с внешними устройствами осуществлялся через программный канал или через канал разрыва данными. Конкретное внешнее устройство, к которому проводилось обращение из программ, определялось селекторным кодом. Канал разрыва данными являлся вырожденным каналом прямого доступа к памяти и позволял считывать или заносить массивы информации в оперативную память машины.

Системное устройство конструктивно было оформлено в виде отдельного модуля. К мини-ЭВМ оно подключалось через общую шину (как и внешнее устройство), а к СУ трех соседних ЭМ через каналы межмашинной связи.

Системные устройства мини-ВС СУММА обеспечивали передачу информации между машинами по способу коммутации сообщений. В состав каждого СУ входили: блок управления (БУ), входной и выходной коммутаторы ($К_1$ и $К_2$), буферный регистр (БР), регистр настройки (РН), блок захвата и приоритетов (БЗП).

Блок управления координировал работу всех остальных схем СУ при реализации его взаимодействий с собственной мини-ЭВМ и соседними ЭМ. В составе БУ имелся 12-разрядный регистр адреса (ГА). Этот регистр использовался для указания ячейки оперативной памяти ми -ЭВМ, к которой осуществлялось очередное обращение СУ. Содержимое РА можно было устанавливать либо из данной мини-ЭВМ, либо из соседней ЭМ. Содержимое РА наращивалось автоматически на 1 после каждого обращения к памяти.

Коммутаторы $К_1$ и $К_2$ системных устройств предназначались для управления потоками информации при межмашинных взаимодействиях в мини-ВС. Входной коммутатор $К_1$ определял направления приема информации из межмашинных каналов, а выходной коммутатор $К_2$ направления передачи в межмашинные каналы. Связь между коммутаторами $К_1$ и $К_2$ осуществлялась через **буферный регистр**.

Регистр настройки представлял собой композицию из четырех регистров: РП, $РМ_1$, $РМ_2$, РГ. Все регистры были программно доступны дня мини-ЭВМ.

- Регистр признаков РП был 13-разрядный, он предназначался для хранения признаков $\{\pi_o, \pi_1, \ldots, \pi_i, \ldots, \pi_{12}\}$. Признаки позволяли задать функции ЭМ при вьшолнении межмашинных взаимодействий.
- Регистр маски $РМ_1$ состоял из трех триггеров T_k^- (k € {1, 2, 3 }), каждый из которых предназначался для маскирования одного из направлении межмашинных связей.
- Регистр маски $РМ_2$ включал в себя также три триггера: T_k^+ (k € {1, 2, 3 }), а его состояние определяло направления приема данных в ЭМ из межмашинных каналов.
- Регистр границ РГ 3-разрядный — использовался при формировании «границ» между машинами, т. е. при разбиении мини-ВС на подсистемы. Содержимое этого регистра определяло направления межмашинных связей ЭМ, из которых запрещался прием любой информации

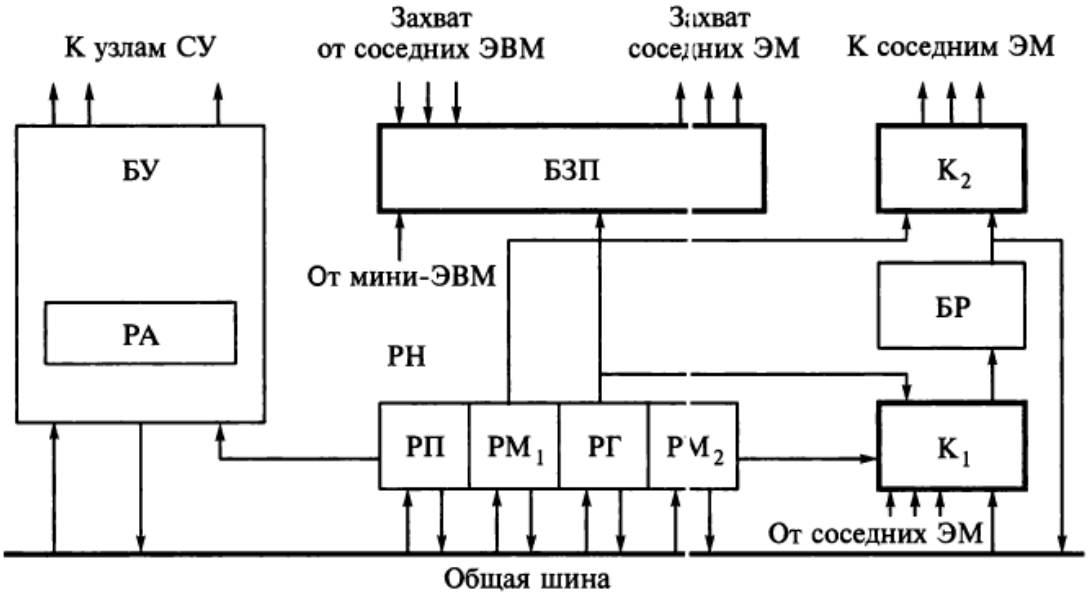


Рис. 7.17. Функциональная структура системного устройства мини-ВС СУММА:

БУ — блок управления; $К_1$, $К_2$ — коммутатор; РН — регистр настройки; БЗП — блок захвата и приоритетов; СУ — системное устройство; ЭМ — элементарная машина; БР — буферный регистр; РП — регистр признаков; $РМ_1$, $РМ_2$ — регистры маски; РГ — регистр границ; РА — регистр адреса

9.5.3. Системные команды мини-ВС СУММА

Системные команды мини-ВС СУММА были разделены на три группы.

Первую группу составляли команды обращения из мини-ЭВМ в собственное СУ

- установить (или сбросить) заявку на обслуживание процессора
- сбросить все заявки на обслуживавание, кроме заявки собственного процессора
- осуществить обмен с регистрами СУ
- сбросить триггер признака готовности СУ
- пропустить очередную команду по значению триггера готовности СУ

Команды второй и третьей групп выполнялись совместно и позволяли осуществить обмен информацией между любыми ЭМ подсистемы. Процесс передачи, инициированный передающей ЭМ, начинался с «захвата» собственного СУ. Применялось два режима захвата мягкий и жесткий. При мягком режиме процессор устанавливал в СУ заявку на обслуживание и ждал освобождения СУ от текущей работы. При жестком режиме захват СУ происходил независимо от текущего состояния СУ. Передача процессором необходимой информации в СУ осуществлялась . лишь после подтверждения, что захват СУ произошел.

Обмен между ЭМ выполнялся 13-разрядными словами, все разряды слова передавались параллельно. Тринадцатый разряд приформировывался СУ к каждому слову, полученному от процессора, и указывал назначение слова. Единичное значение этого разряда со ответствовало передаче кода настройки, а нулевое передаче операнда или адреса: последний посылался в регистр адреса СУ. Любой передаваемый между ЭМ массив информации начинался со слова настройки.

Слово настройки предназначалось для задания функционирования СУ всех машин, принимавших данный массив, и состояло из двух частей

Рис. 7.18. Формат слова настройки мини-ВС СУММА

Указатели:

запроса прерывания (1 соответствовала переходу к подпрограмме обработки прерываний после приема всего массива информации; 0 запрещал переход);

второго слова массива передаваемой информации (1 означала, что второе слово являлось начальным адресом следующего за ним массива; следовательно, оно заносилось в регистр адреса системного устройства; 0 соответствовал операнду, который заносился в память ЭВМ по адресу, определяемому текущим содержимым регистра адреса СУ);

приема кодов (1 настраивала ЭМ на прием операндов или адреса для регистра адреса СУ; 0 запрещал прием);

ретрансляции настройки (1 разрешала ретрансляцию слова настройки в выходные каналы межмашинной связи; 0 не разрешал ретрансляцию);

конца обмена (1 означала: завершить обмен информацией и освободить системные устройства принимающих ЭМ; 0 обмен продолжить);

типа обмена (1 соответствовала быстрому обмену, при котором принимающие ЭМ были либо заняты приемом кодов, либо находились в состоянии ожидания; 0 означал медленный обмен, при котором принимающие ЭМ после завершения приема в память очередного кода, возвращались к своей программной работе).

Идентификатор ЭМ, участвующих в системном взаимодействии, определяемом указателями. Элементарные машины, отмеченные признаком $\pi_i = 1$, i €{1, ... , 12}, осуществляли настройку своих СУ в соответствии со значениями указателей.

Разметка выполнялась той машиной, которая получила очередную заявку на решение задачи (ведущей ЭМ). Ведущая ЭМ создавала компонент связности, включавший в себя все доступные ей машины при данной загруженности системы процессе разметки ведущая. ЭМ передавала во все выходные каналы слово настройки с указателем ретрансляции, равным 1, и с идентификатором π_0 =1. Машины, соседние с ведущей, принимали и одновременно ретранслировали слово настройки в другие ЭМ системы. В результате выполнении описываемого «волнового» алгоритма все исправные ЭМ, доступные ведущей машине, образовывали компонент связности. Компонент связности имел вид дерева, корнем которого была ведущая ЭМ.

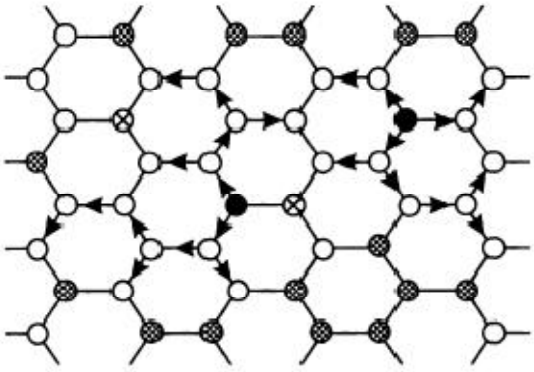


Рис. 7.19. Формирование компонентов связности в мини-ВС СУММА:
● — ведущая ЭМ; ⊗ — неисправная ЭМ; ○ — свободная ЭМ; ◍ — ЭМ, занятая в других подсистемах

9.5.4. Программное обеспечение мини-ВС СУММА

Программное обеспечение мини-ВС СУММА включало в себя супервизор, систему Р-программирования, управляющие системы для автоматизированных систем управления технологическими процессами (АСУТП), комплекс программ технического обслуживания.

Супервизор

Супервизор являлся резидентной программой управления процессами в реальном масштабе времени. Он состоял из подсистем управления процессами и межмашинных взаимодействии.

Подсистема управления процессами

- анализ и обработку заявок от внешних устройств и от объектов управления;

- режим мультипрограммирования в реальном масштабе времени;
- накопление запросов на системные межмашинные взаимодействия
- разрешение коллизии при системных взаимодействиях

Подсистема межмашинных взаимодействии
- реализацию системных взаимодействий
- контроль правильности выполнения системных взаимодействии в мини-ВС.

Система Р – программирования

Система Р-программирования мини-ВС СУММА включала в себя комплекс модифицированных средств программирования мини-ЭВМ «Электроника-100 И»:
- загрузчик для ввода объектных программ в оперативную память мини-ВС
- редактор для приготовления (с клавиатуры пишущей машинки) символических Р-программ
- средства отладки объектных программ;
- библиотеку стандартных параллельных программ, включавшую программы для реализации сложных системных взаимодействий и Р-программы для научно-технических расчетов.

Управляющие системы для АСУТП

Управляющие системы для АСУТП определяли конкретные применения мини-ВС СУММА. Каждая такая система подразделялась на подсистему управления технологическим процессом и банк управляющих программ.

Подсистема управления технологическим процессом обеспечивала следующие виды взаимодействий
- информационное (реализация всех необходимых видов и способов ввода и вывода информации);
- обрабатывающее (выполнение требуемых видов переработки информации: интерполирования, расчета геометрии детали и т. п.);
- управляющее (реализация организационны, технологических и обслуживающих функций по управлению процессом).

Банк управляющих программ включал:
- архив программ управления технологическим процессом (на магнитных лентах);
- оперативную библиотеку программ управления технологическим процессом (на магнитных дисках).

Комплекс программ технического обслуживания

Комплекс программ технического обслуживания мини-ВС СУММА обеспечивал выполнение работ по наладке, контролю и диагностике технических средств. В комплекс входили наладочные, контрольные и диагностические программы.

9.5.5. Области применения мини-ВС СУММА

Система СУММА в 1970-х годах была перспективным вычислительным средством для АСУТП. Для АСУТП, построенных на ее основе, были характерны:

- простота компоновки и настройки на заданный парк оборудования и объектов управления;
- модульная и адекватная наращиваемость вычислительной мощности при развитии производства;
- высокая надежность и живучесть;
- высокая технико-экономическая эффективность;
- длительный срок эффективной эксплуатации (медленное моральное старение).

Применение мини-ВС СУММА было эффективно и при решении широкого класса задач, представленных параллельными программами. Кроме того, она могла быть использована в качестве вычислительного ядра «интегрированных» АСУТП. В таких АСУТП реализовывались функции не
только собственно управления, но и планирования производства, и «проектирования» процесса (например, расчета технологии обработки или расчета поверхностей деталей, если система предназн, мчалась для работ со станками с числовым программным управлением).

Систему СУММА можно было использовать и как автономное средство для решения задач повышенной сложности, а также для моделирования архитектур ВС и параллельных вычислительных технологий.

Функциональная организация СУ позволяла просто адаптировать систему СУММА к конкретным областям ее применения.

9.6. Вычислительные системы семейства МИКРОС

В начале 1980-х годов в качестве базы для построения распределенных ВС с программируемой структурой стали использовать аппаратурно-программные средства микроЭВМ. В Отделе вычислительных систем Сибирского отделения АН СССР проводились работы по научно-исследовательскому проекту МИКРОС*

Архитектура систем семейства МИКРОС:

- MIMD-архитектура
- распределенность средств управления, обработки и памяти
- массовый параллелизм (при обработке данных и управлении процессами)
- программируемость структуры сети межмашинных связен
- возможность программной трансформации MIMD-архитектуры в SIMD и MISD
- децентрализация ресурсов
- асинхронность и близкодействие
- масштабируемость, модульность и однородность

9.6.Вычислительные системы семейства МИКРОС.

В конце 197 0-х годов мини-процессоры вытесняются микропроцессорами, на смену мини- ЭВМ пришли микроЭВМ; создаются параллельные ВС как коллективы микропроцессоров. В начале 1980-х годов в качестве базы для построения распределенных ВС с программируемой структурой стали использовать аппаратурно-программные средства микроЭВМ. В Отделе вычислительных систем
Сибирского отделения АН СССР проводились работы по научно-исследовательскому проекту МИКРОС *, целью которых было создание МИКРОпроцессорных Систем с программируемой структурой (МИКРОС). Результатом работ явилось семейство МИКРОС, включающее модели МИКРОС-1(1986); МИКРОС-2 (1992); МИКРОС-Т (1996).

Архитектура систем семейства МИКРОС:

• MIMD-архитектура;

• распределенность средств управления, обработки и памяти;

• массовый параллелизм (при обработке данных и управлении процессами);

• программируемость структуры сети межмашинных связей;

• возможность программной трансформации MIMD-архитектуры в SIMD и MISD;

• децентрализация ресурсов;

• асинхронность и близкодействие;

• масштабируемость, модульность и однородность.

9.6.1. Функциональная структура ВС МИКРОС.

9.6.2.

Модели элементарных машин ВС. Функциональная структура и состав элементарных машин систем МИКРОС-1 и МИКРОС-2. Функциональная структура
элементарной машины системы МИКРОС-Т. Архитектура транспьютеров семейства Inmos T800. Архитектурные возможности высокопроизводительных микропроцессоров (Intel 860, PowerPC, Alpha).

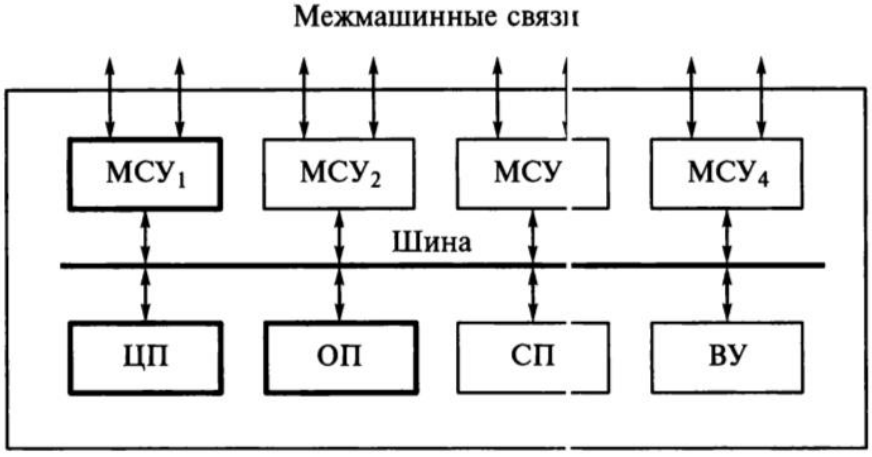


Рис. 7.21. Функциональная структура ЭМ систем МИКРОС-1 и МИКРОС-2:

МСУ — модуль системного устройства; ЦП — центральный процессор; ОП — оперативная память; СП — специальный процессор; ВУ — внешнее устройство

Возможности функциональных структур систем семейства МИКРОС определяются количеством ЭМ, входящих в их состав, конфигурациями ЭМ и топологией сетей межмашинных связей. Количество ЭМ в любой из моделей (МИКРОС-1, МИКРОС-2, МИКРОС-Т) не фиксировано. Каждая ЭМ это многополюсник, число полюсов v в первых моделях систем составляло от 2 до 8, а в модели МИКРОС-Т v = 4. Каждая генерация ВС семейства МИКРОС адекватно учитывала текущие возможности ВТ и интегральной технологии. Для формирования конфигураций ЭМ моделей МИКРОС -1 и МИКРОС-2 использовались средства микроЭВМ отечественного семейства «Электроника». Элементарная машина представлялась композицией из микроЭВМ (вычислительного модуля) и системного устройства (которое, в свою очередь, формировалось из модулей) (рис. 7.21).

Свойством масштабируемости обладали не только модели семейства МИКРОС, но и их ЭМ. Простейшая конфигурация ЭМ состоит из модуля системного устройства (МСУ), центрального процессора (ЦП) и оперативной памяти (ОП). Модуль СУ обеспечивал реализацию системных операций в ВС и непосредственную связь данной ЭМ двумя соседними машинами через полудуплексные каналы. Модуль СУ позволял использовать в качестве каналов различные средства, в частности, экранированные провода (при расстоянии между ЭМ до 30 м), либо радиочастотные кабели (если расстояние между ЭМ не превышало 300 м), либо коммутируемые или выделенные телефонные каналы связи (с использованием аппаратуры передачи данных независимо от расстояния между ЭМ). Заложенная в модуль СУ схема обеспечения связности машин была равно пригодна для формирования как сосредоточенных, так и пространственно распре деленных ВС.

В моделях ВС МИКРОС-1 и МИКРОС -2 в качестве базовых машинбыли использованы микроЭВМ «Электроник 60М» и «Электроника 60-1» соответственно. Технические характеристик" микроЭВМ, точнее их ЦП, отражены в табл. 7.9.

Расширенные конфигурации ЭМ систем МИКРОС-1 и МИКРОС-2 могли иметь до четырех модулей СУ, специальный процессор (СП), дополнительные модули оперативной памяти, набор внешних устройств (ВУ). Специальные процессоры «Электроника МТ-70» или «Электроника 1603» расширяли вычислительные возможности ЦП при решении научно-технических задач, связанных с обработкой значительных массивов данных и с выполнением больших объемов однородных вычислений.

Техническая характеристика	МикроЭВМ	
	«Электроника 60М»	«Электроника 60-1»
Разрядность слов, дв. разр.	16	16
Разрядность чисел с плавающей запятой, дв. разр.	32	32
Объем адресного пространства, К слов	32	128
Максимальная емкость ОЗУ, К слов	28	124
Число команд	81	130
Быстродействие, 10^3 опер./с	250	500
Число уровней прерывания	2	4

Таблица 7.10

Техническая характеристика	Спецпроцессор	
	«Электроника МТ-70»	«Электроника МС 1603»
Разрядность чисел с фиксированной запятой, дв. разр.	16	16
Емкость памяти данных, К слов	32	32–256
Емкость памяти микропрограмм, бит	512 × 56	512 × 32
Число операций над массивами данных	32	32
Время выполнения операций сложения, нс	200	200
Время выполнения операций умножения, нс	400	200
Время выполнения быстрого преобразования Фурье (1024 комплексные точки), мс	30,1	Менее 11

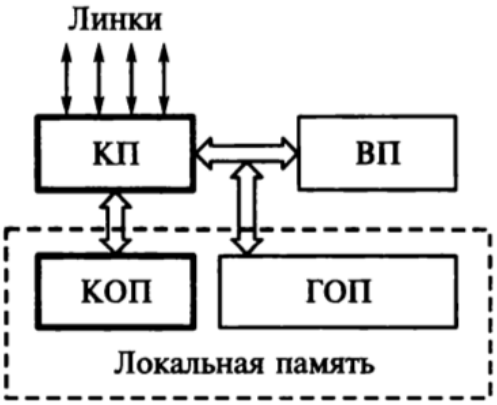


Рис. 7.22. Функциональная структура ЭМ МИКРОС-Т:

КП — коммуникационный процессор; ВП — вычислительный микропроцессор; КОП — коммуникационная память; ГОП — главная оперативная память

Модули системного устройства для системы МИКРОС-2 обладали большими функциональными возможностями, чем в системе МИКРОС-1. Их аппаратура, в частности, позволяла осуществлять: обработку входных/выходных запросов для межмашинных связей (линков); анализ семафоров; формирование пакетов выходных сообщений; управление входными и выходными портами при выполнении системных команд; мультиадресные передачи информации; совмещение межмашинных обменов информацией с вычислениями.

Система МИКРОС-Т базируется на транспьютерных технологиях *. Такие технологии позволяют формировать двумерные ВС с массовым параллелизмом. двумерные структуры ВС формируются путем отождествления полюсов-линков (Link связь).

Простейшая конфигурация ЭМ представляется транспьютером (например, Inmos T 805) с памятью, развитые конфигурации ЭМ могут включать в себя: высокопроизводительные микропроцессоры Intel 860 (компания Intel), PowerPC (альянс компаний IBM, Apple и Motorola), Alpha (компании DEC и Сотраq) и др. Для формирования ЭМ системы МИКРОС-Т могут быть использованы стандартные решения зарубежных и отечественных фирм-производителей транспьютерных модулей.

Транспьютер включает в себя средства для выполнения вычислений (ЦП, АЛУ с плавающей точкой, внутрикристальную память) и 4 канала для связи (линка) с другими транспьютерами и/или другими устройствами. Каждый линк представляет собой 2 однонаправленных последовательных канала передачи информации. Встроенный интерфейс позволяет подключать внешнюю память емкостью до 4 Гбайт (рис. 13.5).

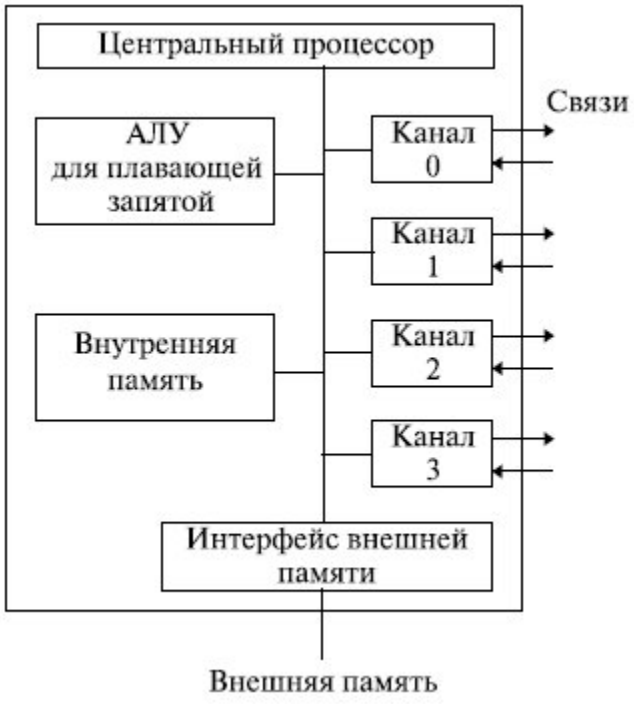


Рис. 13.5. Структура транспьютера

PowerPC спроектирован в соответствии с принципами RISC, в рамках концепции возможна суперскалярная реализация. Существуют версии дизайна как для 32-, так и для 64-разрядных вариантов. Помимо базовых спецификаций POWER, PowerPC обладает:

- возможностью, отсутствующей в PowerPC G5, работать в двух режимах — big-endian и little-endian, переключаясь между режимами во время вычислений;
- однопроходными формами некоторых инструкций для вычислений с плавающей запятой, в добавление к двухпроходным;
- дополнительными инструкциями для вычислений с плавающей запятой, разработанными Кейтом Дифендорфом из Apple;

- обратной совместимостью с 32-разрядным режимом в 64-разрядных версиях;
- отсутствием некоторых особо специфических команд POWER, некоторые из которых могут эмулироваться операционной системой, если понадобятся.

9.6.3. Программное обеспечение МИКРОС

В основу операционной системы МИКРОС положены следующие **принципы:**

• независимость от структуры ВС и числа машин в ней;
• модульность построения;
• распределенность и децентрализованность модулей по машинам ВС;
• локальность связей между модулями;
• асинхронность взаимодействия модулей;
• развиваемость (изменяемость и пополняемость состава модулей, в частности возможность замены программных модулей на аппаратурные);
• иерархичность построения: стратификация системы на уровни, каждый из которых строится на основе предыдущих и освобождает пользователя от специфических для уровня операций по погружению задачи в систему;
• преемственность с ОС базовых микропроцессорных средств (либо микроЭВМ «Электроника», либо транспьютеров, в зависимости от моделей семейства ВС МИКРОС).

Все созданные генерации ОС (МИКРОс-1, МИКРОС-2, МИКРОС-Т) являются распределенными и децентрализованными. Децентрализованная распределенная ОС МИКРОС способна функционировать в ВС произвольной конфигурации; ОС создает в каждой ЭМ «окружение», позволяющее осуществлять динамическую настройку адаптирующейся параллельной программы на существующую конфигурацию ВС (или подсистемы). Децентрализованные процедуры маршрутизации обеспечивают передачу сообщений между любыми ЭМ системы. Указанные свойства ОС МИКРОС являются основой для поддержки живучести ВС (и, следовательно, для организации отказоустойчивых вычислений). Следует подчеркнуть, что ОС МИКРОС-Т имеет иерархическую структуру (рис. 7.24). Каждый уровень ОС строится на основе предыдущих.

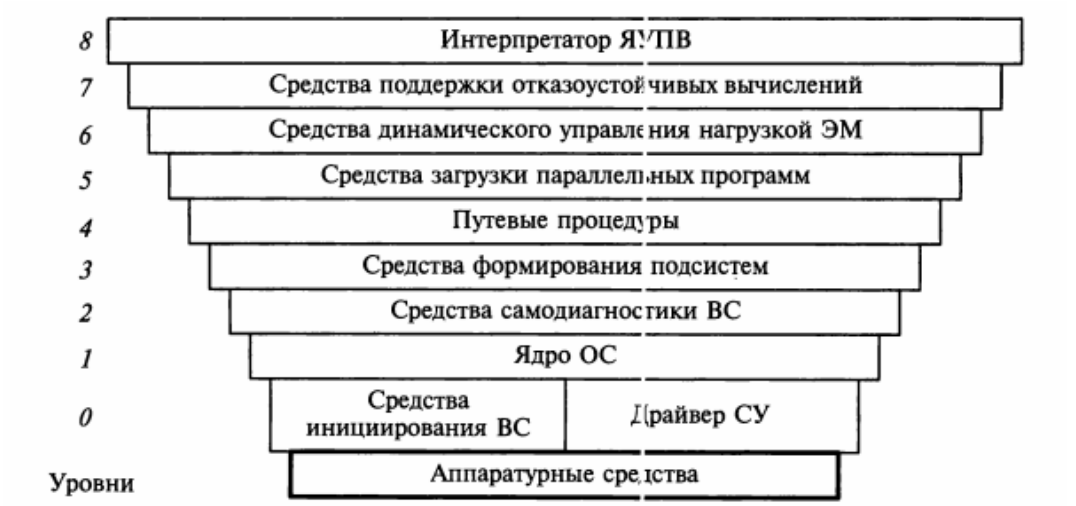


Рис. 7.24. Операционная система МИКРОС-Т

Уровень 2 ОС образован средствами самоконтроля и самодиагностики ВС, которые обеспечивают проверку работоспособности компонентов ВС и локализацию неисправных ресурсов. Самоконтроль и самодиагностика реализуются как параллельные процессы, выполняемые всеми ЭМ системы.

Средства формирования виртуальных подсистем (**уровень 3**) используются в мультипрограммных режимах работы системы. Они выделяют машины, входящие в подсистему, посредством созданных в этих машинах «окружений». Окружение содержит информацию о принадлежности машины подсистеме и другие параметры, в частности задающие вид связности машин: «линейку», «кольцо», «дерево», «решетку» и др. Значения элементов окружения используются путевыми процедурами (**уровень 4**) при выполнении обменов между машинами подсистемы. Набор путевых процедур предназначается для реализации схем межмашинньх обменов в пределах подсистем и всей системы в целом. Фактически путевые процедуры распространяют функции ядра ОС на всю систему.

После формирования окружений в машины подсистемы загружаются предназначенные им программы и данные. Загрузка машин осуществляется с помощью специальных средств **уровня 5**.

Уровни 0-5 (за исключением программы инициирования) составляют резидентную часть ОС, содержащуюся в каждой ЭМ вычислительной системы.

Средства динамического управления нагрузкой ЭМ (**уровень 6**) осуществляют перераспределение программ и данных между ЭМ подсистемы по завершении ее формирования или реконфигурации. На **уровне 7** поддержки отказоустойчивых вычислении выполняются операции, связанные с перезапуском параллельных процессов вычислении с заданных точек возврата. Методы управления нагрузкой и восстановления вычислений могут быть как универсальными, так и специализированными, определяемыми областью применения ВС. Следовательно, **уровни 6 и 7** могут быть включены либо в резидентную часть ОС, либо в загрузочный модуль параллельной программы пользователя.

Интерпретатор языка управления параллельными вычислениями (ЯУПВ, **уровень 8**) по командам с терминала порождает процессы, осуществляющие: генерацию подсистемы необходимого типа («дерево», «линейка», «кольцо» и т. п.) из требуемого числа работоспособных ЭМ; загрузку параллельной программы в сформированную

подсистему и инициирование ее выполнения. Интерпретатор загружается в ЭМ, непосредственно связанную с терминалом.

При разработке средств программирования соблюдались следующие положения:

1) независимость от структуры и числа машин ВС;
2) адекватность между языковыми сред .твами для задания паралле-лизма и синхронизации вычислений и структурной, и функциональной ор-ганизацией ВС;
3) развиваемость (пополняемость как языковыми средствами, так и средствами отладки, анализа программ, распараллеливателями последовательных программ, диалоговыми системами обучения параллельному программированию, пакетами параллельных программ);
4) использование языков высокого уровня как для прикладного, так и для системного программирования;
5) преемственность со средствами последовательного программирования, обеспечение постепенного перехода программиста от привычной для него идеологии последовательного программирования к параллельному программированию.

В версии средств программирования МИКРОС-Т имеются языки параллельного программирования Р-ФОРТРАН и Р-С. Эти языки построены путем расширения соответствующих традиционных языков FORTRAN и С примитивами организации межмашинных взаимодействий и примитивами оценки параметров подсистем, на которых исполняются параллельные программы. Первые позволяют организовать взаимодействия между любыми ветвями программы, вторые дают возможность использовать параметры подсистемы для адаптации программы к текущей конфигурации последней. Это свойство существенно **с двух точек зрения**: простоты организации параллельных вычислений и отказоустойчивости. Реализация данных примитивов основывается на средствах распределенной децентрализованной операционной системы МИКРОС-Т.

9.6.4. Архитектурные свойства систем семейства МИКРОС

Опишем архитектурные свойства ВС семейства МИКРОС.

Класс архитектуры любой модели ВС - это MIMD; допустима трансформация архитектуры MIMD в архитектуру MISD или SIMD путем программной перенастройки системы.

Класс ВС- система с программируемой структурой и с распределенным управлением.

Характер пространственного размещения вычислительных ресурсов - сосредоточенный или распределенный.

Основная функционально-структурная единица вычислительных ресурсов - ЭМ .

Функции ЭМ- традиционные для ЭВМ функции по переработке информации плюс функции, связанные с управлением ВС в целом как коллектива (ансамбля) машин.

Масштабируемость ВС поддерживается аппаратурными средствами (системным устройством либо транспьютером) и программным обеспечением.

Количество N элементарных машин не фиксировано, что обеспечивает принципиально неограниченное наращивание производительности ВС.

Виды структуры сети межмашинных связей - произвольные (нерегулярные) графы.

Рекомендуемые структуры ВС:

• для сосредоточенных ВС: оптимальные D_n- и $L(N, v, g)$-графы, т. е. графы, в которых, в частности, обеспечивается минимум среднего диаметра (задержек при межмашинных передачах информации):

D_n-графы имеют параметрическое описание $\{N; \omega_0, \omega_1, \ldots, \omega_{n-1}\}$, где N — порядок; n — размерность графа, числа ω_k таковы, что две вершины с номерами i и j соединены ребром, если выполняется сравнение $i - j \equiv \omega_k \bmod N$, $i, j \in \{0, 1, .., N-1\}$, $k \in \{0, 1, \ldots, n-1\}$;

$L(N, v, g)$-графы — неориентированные однородные графы, в которых N — число вершин, v — степень вершины (число межмашинных связей для каждой ЭМ), g — обхват (длина кратчайшего цикла в графе);

- **для пространственно распределенных ВС**: в условиях отсутствия жестких технико-экономических ограничений те же структуры, что и для сосредоточенных систем, в противном случае любые структуры реально существующих сетей передачи информации или сетей, обеспечивающих при заданных ограничениях связность вычислительных ресурсов.

Наращиваемость (масштабируемость) **размерности** структуры ВС - v = 1 - 8.

Тип оперативной памяти-распределенная и общедоступная.

Аппаратурно-программная база системы:

• для моделей МИКРОС-1 и МИКРОС-2 средства микромашинной техники и спецпроцессоров семейства «Электроника»;

• для модели МИКРОС-Т транспьютерные средства семейства InmosT800 (компании SGS-Thomson) и средства высокопроизводительных микропроцессоров.

Конфигурации ЭМ:

• для моделей МИКРОС-1 или МИКРОС -2 всевозможные допустимые комплексы на основе микроЭВМ «Электроника 60М» или «Электроника 60-1» (или совместимых с ними других микроЭВМ), которые могут иметь в своем составе, в частности, спецпроцессоры «Электроника МТ-70» и «Электроника 1603»;

• для модели МИКРОС-Т либо один из транспьютеров Т805 или Т800 (простейшая конфигурация), либо транспьютер, один из высокопроизводительных микропроцессоров: Intel 860,PowerPC , Alpha и дополнительная оперативная память (расширенная конфигурация).

Коммуникационные средства ЭМ для реализации функций управления системами:

• МИКРОС-1 или МИКРОС-2 модули СУ (СУ-1 или СУ-2), выполненные на полных платах конструктивов для микроЭВМ «Электроника 60М» или «Электроника 60-1»;

• МИКРОС-Т транспьютер Inmos Т805 (или Т800).

Число коммуникационных средств в одной ЭМ в системах:

• МИКРОС-1 или МИКРОС-2 одно СУ в составе от одного до четырех модулей СУ;

• МИКРОС-Т один транспьютер Inmos Т805 (или Т800).

Программное обеспечение ВС:

• МИКРОС-1 или МИКРОС-2:

— распределенные децентрализованные ОС, являющиеся расширением ОС микроЭВМ «Электроника 60М» или «Электроника 0-1 »;

— языки параллельного программирования P-FORTRAN и P-PASCAL, являющиеся языками семейства микроЭВМ«Электроника», дополненными средствами организации системных взаимодействий.

• МИКРОС-Т:

— распределенная децентрализованная ОС МИКРОС-Т (см. рис. 7.24);

— языки параллельного программирования P-FORTRAN и P-C.

Режимы функционирования ВС:

• монопрограммный, обеспечивающий рцшение сложной задачи, при котором все ресурсы ВС используются для реализации параллельных программ и обеспечения требуемого уровня надежности и живучести;

• мультипрограммные (обработка наборов и обслуживание потоков параллельных задач, разделение «времени и/или пространства» и др.), при которых для решения любой задачи или для обслуживания любого задания используется лишь часть ресурсов системы.

Способы обработки данных в ВС:

• распределенный (параллельный), когда о, инородно расчлененные данные и ветви параллельной программы их обработки рассредоточиваются по ЭМ;

• матричный, при котором программа вычислении размещается в одной или нескольких ЭМ, а данные (однородно) распределяются по всем ЭМ;

• конвейерный, когда сегментированная программа распределяется по машинам предварительно настроенного конвейера (или «кольца» или «линейки») и обеспечивается последовательное «пропускание» данных через все ЭМ конвейера.

Рекомендуемая методика распараллеливания сложных задач – крупноблочное распараллеливание, позволяющее засчет минимизации затрат намежмашинные взаимодействия достичь линейной зависимости производительности ВС от числа ЭМ.

Требуемые уровни производительности, емкости памяти, надежности и живучести ВС достигаются путем подбора количества ЭМ и их состава, выбора структуры сети межмашинных связей, использования широких возможностей системных аппаратурно-программных средств по статической и динамической реконфигурации структуры и по варьированию состава системы.

Области применения ВС:

• традиционные сферы применения ЭВМ и векторных процессов, в которых возросли требования по обеспечению производительности, емкости памяти, надежности и живучести и где целесообразно сохранить совмести-мость вычислительных средств;

• сферы применения, связанные с решением трудоемких задач, таких как сложные задачи физики, механики сплошной среды, аэродинамики, бал-листики, метеорологии, обработки изображений и речевых данных, задачи организации баз знаний, искусственного интеллекта;

• сложные большемасштабные системы, среди которых системы управления энергетическими установками, системы управления динамическими объектами и другие системы, характеризуемые высокой эффективностью, безотказностью, живучестью, развиваемостью, компактностью либо распределенностью своих ресурсов и т. п.

Таким образом, ВС семейства МИКРОС основываются на перспективных принципах обработки информации, строятся из аппаратурно-программных средств микропроцессорной техники, обладают гибкими возможностями по статической и динамической реконфигурации своих структур, позволяют достичь высокой производительности, надежности и живучести в широкой области применения.

Продолжением ряда ВС МИКРОС-1, МИКРОС-2 и МИКРОС-Т являются высокопроизводительные ВС с массовым параллелизмом семейства МВС.

10.1 Понятие о вычислительных сетях. Классификация и свойства вычислительных сетей.

Топология вычислительных сетей. Примеры вычислительных сетей.

Сетевые концепции

СЕТЬ – такое соединение двух или более компьютеров, которое позволяет им разделять ресурсы.

Классификация сетей

В **зависимости от размера географической области**, которую охватывают ВС, они подразделяются на 3 основных типа: локальные (LAN), региональные (MAN) и глобальные (WAN).

Локальные сети (ЛВС) объединяют компьютеры в пределах небольшого физического пространства, например одного или нескольких рядом находящихся зданий. Они обслуживают пользователей, находящихся на расстоянии десятков и сотен метров друг от друга, и число этих пользователей в самом лучшем случае не превышает нескольких тысяч человек. Такие маленькие размеры ЛВС позволяют работать на скоростях взаимодействия 10 Мбит/с и выше. Например, ЛВС Fast Ethernet работает со скоростью 100 Мбит/с. Обмен информацией между устройствами ЛВС происходит с большой интенсивностью. Размер информационных блоков в разных сетях колеблется от 38 бит (сеть Cambridg Ring) до 8 Мбит в сетях с жезловым управлением. В качестве среды передачи данных в ЛВС часто используется кабель типа "витая пара", коаксиальный кабель, оптоволокно.

Региональные сети (РВС) используют технологию глобальных сетей для объединения ЛВС в конкретном регионе, например в городе. Они имеют много общего с ЛВС, например высокую скорость передачи и низкий уровень ошибок в канале связи, но должны обладать возможностью передавать более широкий информационный спектр: не только данные и аудио, но и поддерживать видеообмен. В качестве среды передачи в этом случае часто используется оптоволокно.

Глобальные сети (ГВС) охватывают весь земной шар, для этого используются дополнительные средства коммуникации -спутники, антенны и т. д. Например, услуги сети Интернет доступны по всему миру. В качестве среды передачи данных часто используются телефонные линии, спутниковые системы и наземные микроволновые средства. Отличительной особенностью ГВС являются невысокая скорость передачи и более высокий уровень ошибок передачи.

По способу разделения ресурсов ВС могут быть одноранговыми, клиент-серверными (с выделенным сервером), смешанными.

В *одноранговых сетях* все персональные компьютеры могут предоставлять свои ресурсы в распоряжение всех остальных (серверы). Компьютеры, выступающие как пользователи чужих ресурсов, называются клиентами. Все компьютеры равноправны и в сети отсутствует централизованное управление ресурсами. Типовые одноранговые сети из 5-10 компьютеров строятся под управлением операционных систем Windows 95/98.

Клиент-сервер. Более мощные сети работают в режиме клиент-сервер или с выделенным сервером. В этом случае выделяется специальный компьютер-сервер, который занимается только обслуживанием сетевых запросов. Сервер представляет собой централизованное хранилище сетевых ресурсов и предназначен для централизованного обеспечения безопасности и управления. В отличие от одноранговых, в клиент-серверных моделях обычно требуется один пароль для доступа к сетевым ресурсам. Такие сети называются пользователеориентированными.

Большинство сетей могут быть организованы как смешанные, в них используются как клиент-серверные технологии, так и одноранговые.

Топология сетей. Большое значение для сети имеет топология, которая описывает физическое расположение программно-аппаратных компонентов (физическая топология) и методы для помещения, извлечения и перемещения данных в среде (логическая топология). К ним относятся:

- общая шина (bus);
- кольцо(r^);
- звезда (star);
- сотовая (cellural);
- полносвязная (mesh).

Комбинация этих топологий дает гибридную топологию.

В системах с топологией **общая шина** сетевые адаптеры подключены параллельно к единственному каналу связи - магистрали. *Управление шиной* может быть как централизованное, так и распределенное. При централизованном управлении к шине подключается специальная станция-арбитр, которая регулирует право передачи информации в канал. При распределенном управлении все подключенные станции считаются равноправными и разделяют канал с помощью специальной процедуры - метода множественного доступа. Одной из самых известных сетей с общей шиной является Ethernet фирмы Xerox.

Достоинства: шинная топология представляет собой быстрейший и простейший способ установки маленькой или временной сети. Она требует меньше оборудования, чем остальные сети, и ее легче устанавливать и настраивать.

К *недостаткам* такой топологии следует отнести уязвимость при неполадках в магистральном кабеле и трудность изоляции отдельных станций или других компонентов при неправильной работе.

Для **кольцевых систем** характерно наличие однонаправленного замкнутого канала связи, который разрывается сетевыми устройствами доступа (интерфейсами). Посланное одним интерфейсом сообщение последовательно проходит по кольцу от одного узла к другому, пока не доберется до узла- получателя или не вернется к своему отправителю. В разных сетях удаление такого сообщения-кадра происходит на разных стадиях: кадр может удаляться своим отправителем либо получателем. В некоторых сетях вводят специальную станцию-монитор, которая отлавливает "заблудившиеся" кадры, не нашедшие вовремя своих получателей или отправителей, и уничтожает их.

Топология кольца имеет ряд *недостатков:* его трудно поддерживать и переконфигурировать в больших сетях. Кроме того, неполадки в кабеле фатальны для функционирования всей сети.

Сети со **звездной топологией** имеют в качестве центрального узла концентратор, который как бы тиражирует пришедшее но одной из линий связи сообщение и рассылает его всем остальным станциям сети. Таким образом организуется широковещательная передача. В качестве примера подобных сетей можно привести сеть Fast Ethernet на витой паре со скоростью передачи 100 Мбит/с.

К *достоинствам* таких топологий следует отнести прекрасное масштабирование, независимость работоспособности всей сети от неполадок на отдельной станции или фрагменте кабельной системы, относительная простота расширения сети и ее реконфигури-рования.

Недостатками топологии является необходимость большого количества кабеля, больше, чем при остальных топологиях, и зависимость работоспособности сети от концентратора.

Сети с **топологией в виде сот** определяют принципы беспроводной связи для географических областей, разделенных на ячейки (соты). Каждая ячейка представляет собой часть общей области, внутри которой функционируют конкретные соединения, связывающие устройства с центральной станцией. Центральные станции соединены в виде сетки. В этом случае при пересылке информации существует множество альтернативных маршрутов, что позволяет поддерживать отказоустойчивость сети, оптимизировать нагрузку при передаче и гарантировать минимальную задержку при доставке сообщений.

Полносвязная топология обеспечивает соединение точки к точке всех сетевых устройств между собой. Несмотря на то, что такая топология требует огромного количества кабеля и очень сложна в установке, полносвязные сети обладают повышенной устойчивостью к сбоям!

Следует отметить, что, хотя рассмотренные топологии обладают широким набором теоретических и практических преимуществ, определяющих их быстрое развитие, они не исчерпывают всего многообразия различных компьютерных сетей. Для описания этого многообразия служат гибридные топологии, например звезды на общей шине, звезды на кольце и т. д.

Примеры локальных вычислительных сетей

Сеть шинной топологии - Ethernet

Одной из первых среди ЛВС шинной структуры была создана сеть Ethernet, разработанная фирмой Xerox. В этой сети был применен метод доступа МДКН/ОК. Технология Ethernet наиболее распространена в ЛВС.

Так, по данным на 1996 г. 85 % всех компьютеров в ЛВС были в сетях Ethernet.

В качестве линий передачи данных в ЛВС используют коаксиальный кабель, витую пару проводов. Длины используемых отрезков коаксиального кабеля не должны превышать нескольких сотен метров, а у витой неэкранированной пары проводов — десятков метров. При больших расстояниях в среду передачи данных включают формирователи сигналов — повторители для сопряжения отрезков. Для связи компьютеров со средой передачи данных используют сетевые контроллеры (адаптеры, сетевые карты), управляющие доступом к сети, и приемопередатчики, служащие для связи сетевого контроллера с линией связи.

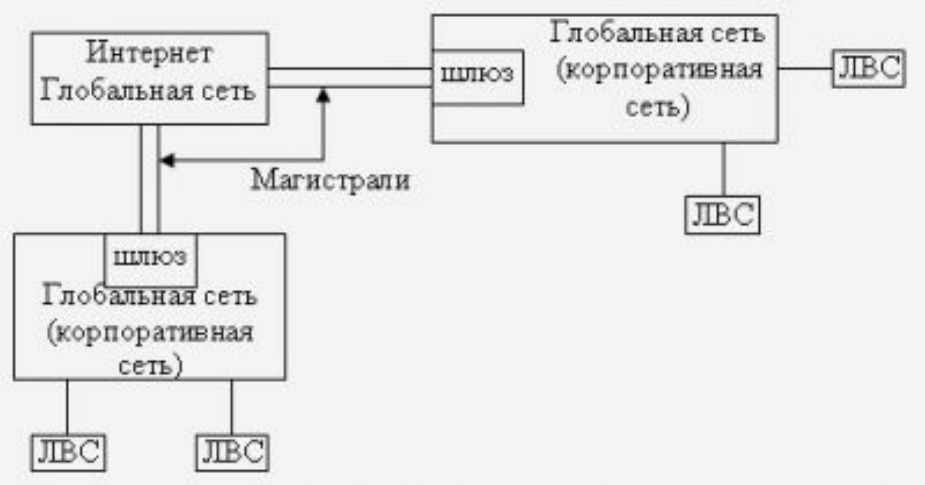
Сеть кольцевой топологии - Token Ring

Из кольцевых ЛВС наиболее распространены сети с передачей маркера по кольцу и среди них: ЛВС типа Token Ring (сеть с таким названием была разработана фирмой IBM).

Типичная реализация сети Token Ring характеризуется следующими данными: максимальное число станций 96; максимальное число концентраторов 12; максимальная длина замыкающего кабеля 120 м; максимальная длина кабеля между двумя концентраторами или между концентратором и станцией 45 м; два варианта скорости передачи данных по линии 4 или 16 Мбит/с; используется маркерный способ.

Сеть Token Ring рассчитана на меньшие предельные расстояния и число станций, чем сеть Ethernet, но лучше приспособлена к повышенным нагрузкам.

10.2 Архитектура Internet



ЛВС - локальная вычислительная сеть

Интернет представляет собой всемирную сеть, информация в которой хранится на серверах. Сервера имеют свои адреса и управляются специальными программами. Они позволяют пересылать почту, файлы, производить поиск в базах данных. Обмен информацией между серверами выполняется по высокоскоростным каналам связи. Доступ отдельных пользователей к ресурсам Internet осуществляется по телефонной сети через провайдера или корпоративную сеть. Провайдером могут быть организации, имеющие модемный пул для соединения с клиентами и выхода в Internet.

В качестве высокоскоростной магистрали используются выделенные телефонные линии, оптоволоконные и спутниковые каналы связи. Для подключения к интернету используется специальный компьютер, который называется шлюзом или gateway. На нем устанавливается программное обеспечение, осуществляющее обработку сообщений, проходящих через шлюз. Каждый шлюз имеет свой IP-адрес. Шлюзы обмениваются между собой информацией о маршрутизации и состоянии сети, используя специализированный шлюзовой протокол. Провайдер имеет свой шлюз в интернет и позволяет пользователям подключаться к сети через этот шлюз. Шлюзы бывают двух типов: внутренние и внешние.

Внутренними называют шлюзы, расположенные в небольшой подсети и обеспечивающие связь с большой корпоративной сетью. Эти шлюзы поддерживают связь между собой с помощью внутреннего шлюзового

протокола IGP - Internal Gateway Protocol.

Внешние шлюзы применяются в больших сетях, настройки их постоянно меняются из-за изменений в мелких подсетях. Связь между внешними шлюзами осуществляется через внешний шлюзовой протокол EGP - External Gateway Protocol.

10.3 Распределенные вычислительные системы. Определение, архитектурные принципы,классификация систем. Примеры реализаций распределенных ВС.

Распределенная ВС - объединение пространственно удаленных друг от друга сосредоточенных ВС, основанное на принципах:

• параллельности функционирования сосредоточенных ВС (т.е. способности нескольких или всех сосредоточенных систем совместно и одновременно решать одну сложную задачу, представленную параллельной программой);

• программируемости структуры (т.е. возможности автоматически настраивать сеть связи между сосредоточенными ВС);

• гомогенности состава (т. е. программной совместимости различных сосредоточенных ВС и однотипности элементарных машин в каждой из них).

Распределенные ВС в общем случае предназначаются для реализации

параллельных программ решения задач на рассредоточенных в пространстве вычислительных ресурсах. Они должны быть приспособленным и и для выполнения функций, присущих вычислительным сетям, и для реализации последовательных программ. В распределенных ВС допустимо централизованное и децентрализованное управление вычислительными процессами.

Примеры реализации: семейство МИКРОС, МИНИМАКС (могла функционировать в составе распределенных ВС),ILLIAC-IV(для сети ARPA).