Введение

Целью курса “Архитектура параллельных вычислительных систем” является рассмотрение способов организации параллелизма обработки информации на различных уровнях вычислительных систем. Будут рассмотрены концептуальные (архитектурные) решения такой организации и структурные решения, позволяющие реализовать предложенные концепции. Следует отметить, что области сущностей вычислительных систем, покрываемые понятиями “архитектура” и “структура”, заметно пересекаются, и в данном курсе не будет преследоваться их строгое разделение там, где в этом нет необходимости.

Понятие архитектуры параллельной вычислительной системы является достаточно широким, поскольку под архитектурой понимается и общая организация параллельной обработки данных, используемая в системе, и особенности системы команд процессоров, и организация памяти, и топология связи между процессорами.

Попытки систематизировать все множество архитектур предпринимались, начиная с 60-х годов прошлого века, и продолжаются сейчас.

Часто используют два понимания термина “архитектура” – в ”узком” и “широком” аспекте.

С термином “архитектура” в “узком” аспекте связываются все те сущности вычислительной системы, которые определяют особенности программирования для нее, в первую очередь, система команд, выполняемых процессорами.

С термином “архитектура” в “широком” аспекте связывается разделение вычислительной системы на уровни с определением интерфейсов между ними (например, уровни организации аппаратуры, функций системного программного обеспечения – операционных систем и систем программирования, прикладного программного обеспечения).

В 1966 г. М.Флинном (M.Flynn) был предложен подход к классификации вычислительных систем, основанный на понятии последовательностей (потоков) команд и данных.

В классификации Флинна описываются четыре архитектурных класса:

OKOD (SISD) - одиночный поток команд и одиночный поток данных;

OKMD (SIMD) - одиночный поток команд и множественные потоки данных;

MKOD (MISD) - множественные потоки команд и одиночный поток данных;

MKMD (MIMD) – множественные потоки команд и множественные потоки данных.

OKOD. К этому классу относятся компьютерные системы, которые имеют один процессор, способный обрабатывать один поток последовательно исполняемых инструкций.

OKMD. В этих системах одна и та же инструкция параллельно выполняется над разными данными.

Подклассом OKMD – систем являются векторные компьютеры. Команды векторных компьютеров оперируют массивами данных подобно тому, как команды “скалярных” машин оперируют отдельными элементами таких массивов.

MKOD. Теоретически в этом классе машин множество потоков инструкций должно выполняться над единственным потоком данных. Реальных машин этого класса создано не было.

MKMD. Эти машины параллельно выполняют несколько потоков инструкций над различными потоками данных. Этими потоками инструкций могут быть программы ветвей (параллельных процессов) одной задачи или программы разных задач.

Подробная организация параллелизма в вычислительных системах классов OKOD, OKMD и MKMD будет являться предметом рассмотрения в представляемом курсе.

Внутрипроцессорный параллелизм

В процессе совершенствования процессоров вычислительных машин было пройдено много этапов развития внутрипроцессорного параллелизма:

- организация параллельной работы основных устройств – арифметико-логического устройства (АЛУ), устройства управления (УУ), оперативной памяти (ОП) за счет введения устройств промежуточного хранения обрабатываемой информации – возникновение конвейера исполнения команд;
- разделение АЛУ на функциональные устройства (ФУ) выполнения отдельных операций (например, сложения, умножения, деления) и организация параллельной работы ФУ;
- организация конвейерного выполнения операций в ФУ;
- организация совмещенного выполнения нескольких потоков команд в обрабатывающем узле процессора за счет чередования выборки команд для выполнения из разных потоков для эффективной загрузки ФУ;
- организация параллельного выполнения нескольких потоков команд в обрабатывающем узле процессора за счет использования в нем нескольких комплектов оборудования (“многоядерность”)

Параллелизм работы основных устройств ЭВМ.

В самых первых ЭВМ основные устройства – ОП, УУ, АЛУ работали последовательно.

Вначале осуществлялся выбор текущей команды программы из оперативной памяти. Устройства УУ и АЛУ простаивали.

Затем выполнялась дешифрация в УУ выбранной команды и определялись адреса нахождения в ОП операндов, необходимых для ее выполнения в АЛУ (или адрес в ОП команды, на которую следует передать управление). ОП и АЛУ простаивали.

На следующем этапе снова работала ОП – выполнялся выбор операндов или первой указанной команды из новой последовательности команд. УУ и АЛУ простаивали.

Затем с использованием выбранных операндов команда выполнялась в АЛУ (если она не являлась командой передачи управления). ОП и УУ простаивали.

Последним этапом было осуществление записи в ОП результата выполненной в АЛУ команды (если запись предусматривалась в команде). УУ и АЛУ простаивали.

После осуществления указанных этапов выполнения команды начинался аналогичный цикл действий для выполнения следующей команды. Простои в работе основных устройств ЭВМ были весьма значительными.

Введение двух новых структурных решений позволило ликвидировать простои в работе основных устройств ЭВМ и обеспечить их параллельную работу. Этими решениями были:

- организация параллельной работы нескольких блоков оперативной памяти (“расслоение” памяти);
- организация буферного хранения информации, передаваемой между основными устройствами ЭВМ.

Одновременно существенному уменьшению количества команд в программе и обращений к оперативной памяти послужило:

- введение в устройстве управления индексных регистров, используемых для автоматической модификации адресов операндов (например, для работы с массивами данных в программных циклах);
- организация автоматического повторного выбора из буферных устройств промежуточного хранения информации (без обращения к оперативной памяти) часто используемых команд и данных – организация “кэш-памяти”.

Уменьшению числа обращений к оперативной памяти способствовало также введение для часто используемых данных регистров, непосредственно указываемых в командах (регистров “общего назначения”).

Реализация параллелизма работы основных устройств ЭВМ удачно осуществлена в отечественной универсальной ЭВМ БЭСМ-6 (1967г.), созданной в Институте точной механики

и вычислительной техники (ИТМ и ВТ) под руководством академика Сергея Алексеевича Лебедева.

Образовавшийся в БЭСМ-6 “конвейер команд” - параллельное выполнение команд на разных стадиях их исполнения (от выборки из ОП до выполнения в АЛУ и записи результата выполнения в ОП) позволил в 4 раза увеличить быстродействие машины по сравнению с использованием последовательной работы ее основных устройств при той же элементной базе и конструкции ЭВМ.

Использование в командах “прямо адресуемых” регистров было реализовано в ряде ЭВМ IBM-360 и затем в производимых в СССР и странах Совета экономической взаимопомощи (СЭВ) ЭВМ “единой серии” (ЕС ЭВМ).

Множественность функциональных устройств процессора

Разделение универсального арифметико-логического устройства (АЛУ), выполнявшего широкий набор операций, на набор устройств выполнения отдельных операций или групп “родственных” операций (сложения-вычитания, умножения, деления, сдвига, операций алгебры логики и т.п.) – «функциональных» устройств позволило организовать их параллельную работу. Получившееся при этом заметное увеличение объема оборудования должно было компенсироваться существенной эффективностью такого параллелизма. Для этого в идеале нужно было обеспечить параллельную подачу команд для выполнения на этих устройствах и готовность операндов, необходимых для выполнения команд.

Задержки при последовательной подаче из УУ на ФУ независимых по операндам “векторных” команд работы с массивами данных (“векторами”) при большой длине массивов не сильно сказываются на возможности параллельного выполнения этих команд (при этом, конечно, должны быть готовы операнды).

В случае последовательности “скалярных” команд эффективность параллелизма работы ФУ может быть достигнута только за счет одновременной загрузки ФУ этими командами. Это реализуется за счет выбора из ОП “длинного командного слова”, содержащего набор команд, соответствующий имеющемуся в процессоре набору ФУ. С определенными ограничениями этот подход реализуется в современных «скалярных» процессорах.

Конвейерное выполнение операций

Одновременно с организацией параллельной работы функциональных устройств удалось достичь ускорения работы каждого функционального устройства в несколько раз за счет разделения выполнения операции в ФУ на автономные этапы (“ступени”) и организации последовательного выполнения этих этапов в конвейерном режиме (это, естественно, потребовало некоторого увеличения оборудования ФУ). Принимая время выполнения каждого этапа операции за такт (для многих процессоров это реально так) и принимая на первую ступень в каждый такт данные для выполнения очередной операции, после выполнения всех этапов первой операции и получения ее результата результаты остальных операций получаются на выходе ФУ каждый такт. Таким образом в ФУ осуществляется параллелизм выполнения нескольких операций (по числу этапов выполнения операции) на разных этапах (стадиях, “ступенях”) их выполнения.

Эффективное использование конвейерных ФУ достигается при наличии каждый такт операндов для выполнения операции. Это возможно при предварительной (до выполнения операции) загрузке их (как правило, массивов данных – “векторов”) из ОП в группу специальных регистров и последующем приеме их из регистров на вход конвейерных ФУ без задержки. Возможен и прием каждый такт операндов из ОП за счет необходимого для этого “расслоения” памяти.

При разделении ФУ на K ступеней выполнение N операций на нем будет осуществлено за K+(N-1) тактов.

Примером конвейерного выполнения операций в ФУ может служить четырехступенчатый конвейер выполнения операции умножения чисел, представленных в двоичной системе счисления с “плавающей запятой”. На первой ступени конвейер выполняется сложение “порядков” операндов (степеней двойки – основания системы счисления), на второй ступени выполняется перемножение мантисс операндов. На третьей ступени выполняется “нормализация” результата (приведение значения мантисс в установленный диапазон значений с соответственным изменением порядка результата), на четвертой ступени – округление результата.

В случае выполнения нескольких связанных “векторных” операций возможна организация их конвейерного выполнения без запоминания на регистрах или в ОП промежуточных результатов. Результаты выполнения операций будут поступать с выходов одних ФУ на входы ФУ, выполняющих операции, связанные с предшествующими. Фактически образуется длинный конвейер с числом ступеней, равным сумме количеств ступеней во всех связанных таким образом ФУ. Такая организация называется “зацеплением” ФУ. Соответственно дополнительно многократно увеличивается производительность процессора при выполнении последовательности связанных векторных операций (достигается так называемая “супервекторная” производительность). Например, при выполнении векторного оператора **E =(A+B) x C/ D** реализуемого последовательностью трех векторных операций на ФУ, имеющих по 4 ступени, работающие за 1 такт, общее время вычисления оператора **E** будет равно
12+(N-1), где **N** – число компонент векторов **A,B,C,D** и **E** То есть, для векторов, содержащих по 64 компонента, данное вычисление, требующее выполнения 192-х двуместных операций, будет выполнено всего за 75 тактов.

Удачным примером организации “векторно-конвейерной” ЭВМ является ЭВМ “Cray-1” (1976 г.), созданная компанией Cray Research руководством Сеймура Крея (эту машину называют первым суперкомпьютером в мире).

Представляет также интерес организация векторно-конвейерной ЭВМ без регистров промежуточного хранения «векторных» данных (ЭВМ Cyber-205 фирмы CDC и созданная в ИТМ и ВТ под руководством Андрея Андреевича Соколова отечественная ЭВМ МКП (“модульный конвейерный процессор”), объединяющая архитектурные решения ЭВМ Cray-1 и Cyber-205 с организацией многопоточных вычислений (последнее будет рассмотрено также в следующем разделе курса).

Совмещение выполнения нескольких потоков команд. Многоядерность.

Особенности последовательности команд, реализующей некоторый алгоритм обработки данных, могут не позволить использовать параллельно весь набор функциональных устройств процессора (даже после машинно-зависимой оптимизации программы, выполненной транслятором).

Вероятность параллельного использование ФУ в такой ситуации увеличивается при обеспечении в любой момент возможности взятия команд на выполнение из других последовательностей команд, предназначенных к выполнению на этом же процессоре. Это могут быть, в первую очередь, программы различных ветвей (параллельных процессов) решения одной задачи (Hyper – Threading).

Размещение в одном кристалле микропроцессора нескольких комплектов оборудования для выполнения операций (многоядерность) при сохранении использования общей оперативной памяти (возможно и кэш-памяти) позволяет естественно увеличить многопоточность обработки данных в процессоре. Забота о назначении потоков команд для выполнения на различных ядрах процессора, также как и об организации указанного выше режима Hyper-Threading, ложится на операционную систему ЭВМ.

Многопроцессорность.

Параллельное выполнение операций в OKMD – системах.

Для многих задач (аэрогидродинамики, физики твердого тела, геологоразведки, обработки телеметрии и др.) характерно выполнение одинаковых вычислений над различными группами данных (например, вычисления в узлах “сеток” при решении дифференциальных уравнений). Для реализации таких вычислений создавались вычислительные системы также “узловой” топологии – в узлах этих систем размещались указанные группы данных и оборудование для их обработки (арифметико-логические устройства – АЛУ).

Из одного потока команд текущая команда (через общее устройство управления - УУ) передавалась во все узлы системы для одновременного выполнения над имеющимися там данными, то есть обеспечивалась работа OKMD – системы. Наличие в каждом узле системы лишь АЛУ и оперативной памяти (ОП) не позволяет считать аппаратуру узла “полноправным ” процессором, хотя в ряде случаев в литературе такое название используется.

Широко известным историческим примером такой OKMD – системы является разработанная в Иллинойском университете система ILLIAC – IV (1971 г.), в которой 64 вычислительных узла, описанных выше, параллельно работали и объединялись в топологии “решетка” для передачи данных друг другу. Данные в память узлов передавались из универсальной ЭВМ “Burroughs”. Из нее же передавалась в ILLIAC-IV программа вычислений.

Следует отметить отечественную вычислительную систему этого класса ПС-2000, использовавшуюся для решения задач геологоразведки и обработки телеметрии.

MKMD – системы.

Класс MKMD – систем состоит из двух подклассов: подкласс многопроцессорных вычислительных комплексов (МВК) и подкласс многомашинных вычислительных комплексов (ММВК).

Многопроцессорные вычислительные комплексы.

Рассмотренные выше OKMD – системы плохо приспособлены для параллельного выполнения различных потоков команд (разных программных ветвей одной задачи или программ разных задач). Такое выполнение стало осуществляться на нескольких параллельно работающих “полноправных” процессорах (содержащих УУ, АЛУ), использующих общую для них память или имеющих доступ лишь к собственной оперативной памяти. В последнем случае взаимодействие программ, выполняющихся на процессорах, осуществлялось за счет передачи (с поддержкой операционной системы) сообщений друг другу по коммуникациям, связывающим такие вычислительные узлы.

МВК с общедоступной памятью

Под организацией общедоступной оперативной памяти МВК понимается возможность доступа за данными к любой ее ячейке при выполнении операций в любом процессоре комплекса.

Объединение процессоров на общей оперативной памяти с обеспечением одинакового времени доступа от всех процессоров определяет подкласс SMP (Symmetric multiprocessor). Преимуществом такой организации является простота программирования задач даже с учетом необходимости синхронизации и взаимного исключения программ ветвей (параллельных процессов) этих задач. Недостатком такого подхода является неэффективность подключения к общей памяти большого числа процессоров из-за возрастающих задержек выполнения

операций в них в связи с увеличением, несмотря на значительное “расслоение” оперативной памяти, числа конфликтов при обращении к ней и, соответственно, времен ожидания данных при одновременном обращении к блокам памяти от многих процессоров. Объединение на общей памяти даже нескольких десятков процессоров становилось нерациональным.

Примером SMP – комплекса является отечественная вычислительная система “Эльбрус-2”, в котором через матричный коммутатор осуществляется эффективный доступ к памяти десяти процессоров. Через этот же коммутатор осуществлялось подключение к общей памяти специальных процессоров ввода-вывода, через которые осуществлялось подключение внешних устройств.

Другим вариантом использования общедоступной памяти для процессоров является ее распределение по многим вычислительным узлам, объединенных общей магистралью передачи данных, через которую процессор в каждом вычислительном узле при выполнении операций может иметь доступ к ячейкам памяти других вычислительных узлов. При этом время обращения к памяти увеличивается с удалением от процессора того вычислительного узла, к памяти которого происходит обращение. Такая организация вычислительной системы с общедоступной памятью носит название NUMA (Non Uniform Memory Access). Если при выполнении программы процессорами основное количество обращений за данными будет выполняться к памяти “своего” узла или близко расположенных узлов, то использование NUMA-системы будет эффективным при значительном числе вычислительных узлов. Достижение этого требует существенного усложнения программирования вычислений. В противном случае “масштабирование” NUMA –системы будет ограничено. Примерами NUMA-систем являются системы SGI Origin2000, Sun HPC 10000, IBM/Sequent NUMA-Q 200

Имеются вычислительные системы, содержащие в узлах NUMA – системы SMP – подсистемы (SMP - “кирпичи”). Примером такой системы является комплекс SPP-1200, где SMP–подсистемы (“гиперузлы”) объединяются в топологии “двойной тор”.

МВК с разделенной памятью.

Если в каждом вычислительном узле использовать только “собственную” оперативную память, то не будет ограничений на “масштабирование” такой системы и, соответственно, на количество параллельно выполняемых ветвей (процессов) решаемой задачи, выполняемых в узлах системы. В то же время общее время решения задачи может существенно зависеть от эффективности передачи данных между ветвями (процессами) через объединяющую процессоры коммуникационную систему.

Естественно, что размещение программ и данных задачи по вычислительным узлам и организация передач данных (поддерживаемых операционной системой) между узлами вызывает определенные затруднения при программировании решения задачи.

Описанные системы образуют подкласс MPP (Massively Parallel Processor). Многие из них содержат тысячи и десятки тысяч процессоров. Имеется много различных топологий объединения процессоров в MPP- системах: “линейка”, “ кольцо”, “решетка”, многомерные “торы”, “гиперкуб”, “полный граф” соединений и др.

Представляет интерес сравнение двух последних указанных топологий: гиперкуба и полного графа соединений. Гиперкуб ранга n имеет число вершин (вычислительных узлов), равное 2 в степени n. Отношение числа связей (объемов оборудования) гиперкуба к полному графу при одинаковом числе узлов при росте величины n определяет существенное преимущество гиперкуба. Максимальная длина “транзита” данных в гиперкубе (число проходимых связей при передаче сообщения) равна n (в полном графе она всегда равна 1). Эта величина (соответственно и время передачи данных) растет не столь быстро по сравнению с ростом объема оборудования для полного графа. Поэтому во многих случаях реальные MPP – системы строились по топологии гиперкуба.

Примерами MPP–систем являются системы IBM RS/6000 SP2, Intel PARAGON/ASCI Red, CRAY T3E, Hitachi SR8000.

Успешным оказалось построение фирмой IBM MPP–системы с использованием центрального коммутатора (система IBM RS/6000 SP2). К нему подсоединяются до 128 вычислительных узлов, в качестве которых использовались рабочие станции RS6000 (без внешних устройств и, соответственно, оборудования связи с ними или с полным комплектом оборудования для обеспечения ввода информации в систему и вывода результатов вычислений). Такое гибкое решение позволяло также реализовать в системе с одним коммутатором как многопроцессорные, так и многомашинные подсистемы.

Другим интересным примером организации MPP – системы являются созданные под руководством академика Левина Владимира Константиновича отечественные системы МВС-100 и МВС-1000 (различаются лишь типом используемого процессора), вычислительные модули которых состоят из 16 узлов, соединяемых по топологии “решетка” с дополнительными двумя связями между противоположными “угловыми” узлами. Каждый узел содержит основной вычислительный процессор и его оперативную память, а также “связной” процессор с собственной памятью, обеспечивающий в узле прием-передачу данных. Свободные связи узлов используются для соединения вычислительных модулей друг с другом и подключения к системе «внешних» компьютеров.

Решение проблемы ввода-вывода информации в многопроцессорных вычислительных системах любого типа (SMP, NUMA, MPP) осуществляется за счет подключения к системе внешних (дополнительных) компьютеров, обеспечивающих выполнение этой функции, с соответствующим согласованием работы их системного программного обеспечения с системным программным обеспечением мультипроцессора.

Следует отметить, что одной из первых в мире MPP–систем была разработанная в Киеве в Институте кибернетики им. В.М.Глушкова “макроконвейерная ЭВМ” (ЕС-2701), в которой вычислительные узлы на базе стандартных процессоров ЕС ЭВМ (на них выполнялись основные вычисления) объединялись через систему коммутаторов с так называемыми специальными “логическими” узлами, в которых выполнялись части программы решения задачи, обеспечивающие управление процессом ее решения.

В MPP-системах Cray T3E и Cray T3D процессорные узлы объединены в топологии трехмерного тора. Каждая элементарная связь между двумя узлами - это два однонаправленных канала передачи данных, что допускает одновременный обмен данными в противоположных направлениях.

MPP-системы строятся и на базе векторно-конвейерных процессоров (параллельные векторные системы - PVP). К этому подклассу относится линия векторно-конвейерных компьютеров CRAY: CRAY J90/T90, CRAY SV1, CRAY X1, системы NEC SX-4/SX-5, серия Fujitsu VPP.

В некоторых MPP–системах виртуальная память, предоставляемая задаче и используемая процессором вычислительного узла лишь в физической памяти этого узла, может быть отражена в оперативных памятях многих узлов. При возникновении в каком-либо узле потребности в конкретной части виртуальной памяти задачи эта часть передается по внутрисистемным коммуникациям в физическую оперативную память данного узла.

Кластерные вычислительные системы.

Стремление обеспечить высокую величину отношения производительности вычислительной системы к ее стоимости сделало актуальной задачу осуществления MPP – подхода с использованием более дешевого серийного вычислительного и коммуникационного оборудования. При этом вычислительным оборудованием узла может быть и обычный персональный компьютер, а коммуникационным оборудованием широко выпускаемые промышленностью сети Fast/Gigabit Ethernet, Myrinet, InfinyBand, SCI и др. Вместе с набором программного обеспечения параллельной работы узлов такие системы стали называть «кластерными». Стремление получить большой экономический эффект привело к созданию многих типов кластеров и их широкому распространению. В то же время эффективное

решение появляющихся новых типов задач, в том числе задач с нерегулярным обращением к ячейкам памяти очень большого объема, требует появления новых архитектурных и структурных решений в построении вычислительных систем и их реализации в разработках новых “заказных” систем.

Организация памяти вычислительных систем.

Совершенствование организации памяти вычислительных систем всегда было связано с достижением более высокой производительности работы системы, развитием параллелизма обработки данных.

Основными проблемами в организации памяти являются организация иерархии ее уровней и предоставление для задачи виртуальной памяти, отображающейся на различные компоненты и уровни физической памяти.

Иерархия памяти

Иерархия уровней памяти используется для размещения часто используемой информации (программ и данных) в более быстродействующих (“верхних”) уровнях памяти с возможностью в случае необходимости передачи в них информации из менее быстродействующих уровней существенно большего объема. Результаты обработки информации, которые подлежат хранению на более длительный срок, также размещаются менее быстродействующей (“нижних”) уровнях.

При длительной обработке информации и, естественно, ее изменении в верхних уровнях памяти (в которых она не сохраняется при отключении электропитания) желательным является ее периодическое сохранение в нижних уровнях (в которых осуществляется запись на магнитные носители). Наличие одинаковой информации на различных уровнях памяти называется “когерентностью” памяти.

Основными уровнями в иерархии памяти являются:

- уровни быстродействующей памяти относительно небольшого объема по сравнению с объемом следующего основного уровня – оперативной памяти. Как правило, используются наборы “сверхбыстродействующих регистров” для выбора данных из них (для выполнения операций в процессорах) и записи данных в них (результатов выполнения операций в процессорах) практически без задержки. Информация на эти уровни при необходимости передается из следующего основного уровня существенно большего объема – оперативной памяти (или передается в обратном направлении). Сверхбыстродействующие памяти и оперативная память физически реализуются на интегральных схемах, но быстродействие этих схем разное. Объемы указанных уровней памяти изменяются в обратной пропорциональности по отношению к изменению их быстродействия.
- уровень оперативной памяти физически эффективно используется при разделении ее на набор параллельно работающих блоков (“расслоение ” памяти).

Эффективное использование оперативной памяти для информации нескольких задач, выполняемых в многозадачном (многопрограммном) режиме реализуется за счет отображения на нее “виртуальных” памятей этих задач. При этом при отсутствии места в оперативной памяти для такого отображения приходится использовать для размещения виртуальной памяти задач и “нижний” уровень иерархии памяти (магнитные диски).

- уровень памяти на магнитных дисках может иметь несколько каналов (“направлений”) считывания/записи информации и по несколько устройств, подключаемых к каждому из этих каналов.

В ряде случаев вводится промежуточный уровень “массовой” памяти, располагаемый между оперативной памятью и памятью на магнитных дисках, для размещения в нем “активных” файлов, используемых в задачах, требующих большого числа передач данных из этих файлов (и записи в них) и осуществляющих относительно небольшой объем их обработки процессором. Массовая память большого объема реализуется, естественно, на более

медленных интегральных схемах, чем оперативная память. Уровень массовой памяти помогает сгладить дисбаланс между высоким быстродействием процессора и относительно небольшой скоростью работы магнитных дисков. Естественно при этом стремиться к обеспечению когерентности массовой памяти и памяти на магнитных дисках.

В связи с существенным увеличением объема памяти на магнитных дисках использование памяти на магнитных лентах резко сократилось. В некоторых вычислительных центрах осуществляется периодическое (например, один раз в сутки) сохранение изменившихся файлов с магнитных дисков на “магнитные катушки”.

Сверхоперативная память

Наборы регистров, образующих такую память, бывают двух назначений:

- программируемые в командах “прямоадресуемые” регистры (регистры “общего назначения”), предназначенные для размещения в них данных, часто используемых процессором;
- группы регистров, в которых автоматически (аппаратно) сохраняются часто используемые данные (“кэш (cache) - памяти”).

Использование одного типа регистров в процессоре не исключает использования другого типа регистров.

Кэш-память

В кэш-памяти могут размещаться команды и обрабатываемые данные. Информация располагается в кэш-памяти блоками одинаковой длины. Этими же блоками происходит перемещение данных между оперативной и кэш-памятью.

В командах, выполняемых процессором, указываются лишь адреса данных по оперативной памяти. Преобразование этих адресов в адреса кэш-памяти происходит аппаратно (определяются номер блока кэш-памяти и место расположения в нем требуемых данных). Основными аспектами организации кэш-памяти являются:

- способ доступа к кэш-памяти при обращении к находящимся в ней данным из процессора и при перемещении данных в кэш-память из оперативной памяти;
- способ выполнения записи данных в кэш-память из процессора.

По способу доступа к кэш-памяти различаются следующие организации кэш-памяти:

- полностью ассоциативная кэш-память;
- кэш-память с “прямой” адресацией;
- частично ассоциативная кэш-память.

В случае полностью ассоциативной кэш-памяти требуемые данные могут располагаться в любом ее блоке. Нахождение блока происходит путем сравнения требуемого адреса данных по оперативной памяти с находящимися в специальных регистрах при каждом блоке кэш-памяти начальными адресами групп данных, перемещенных в эти блоки из оперативной памяти, с учетом длины блока кэш-памяти. При нахождении блока кэш-памяти определяется и смещение в нем для выполнения непосредственного обращения к данным.

При не нахождении блока кэш-памяти с нужными данными (“промах” при обращении в кэш-память) эти данные считываются из оперативной памяти в некоторый блок кэш-памяти, определяемый по выполняемому аппаратурой алгоритму замещения данных в кэш-памяти. Если данные в блоке кэш-памяти, выбранном для считывания в него данных из оперативной памяти, изменялись процессором, и это изменение не было отражено в оперативной памяти, то предварительно происходит перепись этих данных из выбранного блока кэш-памяти в оперативную память.

Противоположной по отношению к организации полностью ассоциативной памяти является организация кэш-памяти “с прямой адресацией”. В данном случае заранее определено, из каких диапазонов адресов оперативной памяти (длина диапазонов совпадает с длиной блоков кэш-памяти) данные могут быть помещены в конкретный блок кэш-памяти.

Например, при длине блока кэш-памяти 100 (в восьмеричной системе счисления) и количестве блоков 4 в “нулевой” блок кэш-памяти могут быть переданы данные из ячеек оперативной памяти с восьмеричными адресами:

0000 – 0077, 0400 – 0477, 1000 – 1077, 1400 – 1477 и т.д.

В первый блок кэш-памяти могут быть переданы данные из ячеек оперативной памяти с адресами:

0100 – 0177, 0500 – 0577 и т.д.

Для приведенного примера данные с адресом 0432 могут быть найдены лишь в нулевом блоке кэш-памяти при нахождении в указанном специальном регистре при этом блоке адреса 0400, а данные с адресом 0543 могут быть найдены лишь в первом блоке кэш-памяти при нахождении в его специальном регистре адреса 0500.

В случае “промаха” при обращении в определенный блок кэш-памяти выполняется замена информации в нем на требуемую информацию из оперативной памяти (с возможной предварительной записью находившейся в нем информации в оперативную память как и в случае полностью ассоциативной памяти).

Организация частично-ассоциативной памяти обеспечивает разделение блоков кэш-памяти на группы по одинаковому числу блоков по принципу организации кэш-памяти «с прямой адресацией», а поиск данных в выбранной группе блоков кэш-памяти осуществляется по принципу организации полностью ассоциативной памяти. Так для указанного выше примера четырех блоков кэш-памяти возможна организация частично-ассоциативной памяти из двух групп блоков по два блока в каждой. В “нулевую” группу попадают нулевой и первый блоки, а в первую – второй и третий блоки. Данные с адресами 0000 – 0177, 0400 – 0577, 1000 – 1177, 1400-1577 могут находиться лишь в блоках нулевой группы. Данные с указанными выше адресами 0432 и 0543 могут находиться или отсутствовать либо в нулевом, либо в первом блоке кэш-памяти.

Замена информации в частично-ассоциативной кэш-памяти при “промахе” обращения в нее происходит аналогично замене информации в других организациях кэш-памяти.

Естественно, что наиболее высока вероятность нахождения требуемых данных в более сложной по организации полностью ассоциативной кэш-памяти.

Память с прямой адресацией отличается наибольшей простотой нахождения места расположения данных в ней.

Второй основной проблемой использования кэш-памяти является проблема осуществления записи в нее для многопроцессорных вычислительных систем с общей оперативной памятью и отдельными кэш-памятями у каждого процессора (для однопроцессорных ЭВМ с кэш-памятью для выполнения записи производятся уже описанные выше действия).

Поскольку в вычислительных системах с общей памятью данные из некоторой ее области могут находиться в блоках разных кэш-памятей, изменение этих данных при выполнении записи по некоторому адресу оперативной памяти в одной кэш-памяти не должно привести к использованию “старых” значений при обращении за ними по этому же адресу оперативной памяти в другой кэш-памяти. Это реализуется двумя основными способами.

Первый способ заключается в определении нахождения данных с адресами по оперативной памяти, совпадающими с адресом по оперативной памяти записываемых данных в некоторую кэш-память, в блоках других кэш-памятей. В случае нахождения таких данных в блоках других кэш-памятей содержимое этих блоков объявляется недействительным (при каждом блоке кэш-памяти имеется одноразрядный регистр, указывающий действительность или недействительность содержимого блока). Одновременно выполненная запись в кэш-память осуществляется также и в оперативную память (содержимое блока кэш-памяти, в который произведена запись, становится когерентным содержимому соответствующей области оперативной памяти). Если затем для другого процессора потребуются данные по этому же адресу оперативной памяти, они будут прочитаны из оперативной памяти в некоторый блок кэш-памяти процессора, запросившего данные, поскольку непосредственно из

этой кэш-памяти в связи с проведением указанной выше блокировки эти данные не смогут быть считаны. Таким образом другие процессоры в случае необходимости смогут использовать лишь “новые” значения данных, записанные в оперативную память каким-либо процессором.

Второй способ обеспечения когерентности кэш-памятей заключается в выполнении требуемой записи данных в те блоки всех кэш-памятей, в которых отображены данные из одной области оперативной памяти.

Естественно, что для осуществления обоих способов обеспечения когерентности требуется весьма сложная аппаратура, с помощью которой выполняются указанные выше действия.

Виртуальная память

В первых ЭВМ программирование велось с использованием физических адресов оперативной памяти. Каждая задача программировалась с использованием одних и тех же адресов, то есть на машине в каждый период времени могла выполняться лишь одна задача (фактически первые ЭВМ были машинами “на одну персону”, имея в виду программиста или оператора, работавшего на машине). Размер задачи (объем ее программ и данных) во многих случаях практически сразу стал существенно превосходить размер оперативной памяти машины. Пришлось программировать отдельные части (блоки, модули) задачи на одни и те же адреса памяти и вызывать их в программе на эти адреса оперативной памяти из внешней памяти (на магнитных барабанах, магнитных лентах).

Такие действия, называемые “перекрытием программных модулей” (и модулей данных) в памяти или “попеременной загрузкой” модулей (overlay) вызывали большие затруднения при программировании.

Шагом вперед в организации оперативной памяти стало введение аппаратуры, обеспечивающей использование информации задачи на любом месте оперативной памяти. Использовался “регистр базы”, на который помещался начальный адрес размещения задачи в оперативной памяти, и регистр длины (размера) информации задачи. Это позволило программировать задачи на одни и те же адреса, называвшиеся “относительными”, “логическими”, “математическими” и, наконец, “виртуальными” адресами, а размещать (с помощью операционной системы) информацию задачи на любом месте оперативной памяти. Тем самым обеспечивалась возможность помещения в память одновременно информации многих задач и выполнения этих задач в многозадачном (многопрограммном) режиме. Эффект использования такого режима достигается за счет более полной загрузки процессора. Когда временно (например, из-за выполнения обмена данными между оперативной и внешней памятью) не может выполняться одна задача, тут же начинает выполняться другая задача. Поскольку информация задачи занимает непрерывный диапазон адресов, использование регистра длины информации задачи позволяет не допускать обращения в задаче в невыделенную для нее область памяти и тем самым обеспечивать защиту памяти других задач.

К сожалению, такая организация памяти (задаче предоставлялся ограниченный “линейный” участок оперативной памяти) не избавляла программиста от заботы об организации упоминавшегося перекрытия программных модулей в его таким образом организованной “виртуальной” памяти. Рассмотренный способ организации использования оперативной памяти (с достаточно простой аппаратной “поддержкой”) имеет обобщающее название – память “односегментного отображения”.

Освободить программиста от организации перекрытия программных модулей в памяти и обеспечить при программировании понимание возможности использования виртуальной памяти задачи практически любого объема позволило введение поддержанной аппаратно структурированной виртуальной памяти задачи.

Видами такой организации памяти стали: “сегментная”, “страничная” и “сегментно-страничная” организации.

Сегментная организация памяти.

Для каждой задачи память сегментной организации предоставляет возможность разместить в ней сегменты разной длины, содержащие информацию задачи. В этих сегментах находятся законченные программные объекты – модули процедур и модули данных. При программировании предполагается, что все эти объекты находятся на одном уровне структурированной виртуальной памяти задачи и в программе решения задачи не требуется осуществлять их перемещение между оперативной и внешней памятью (это перемещение будет выполнять операционная система). То есть виртуальная память задачи структурирована как набор модулей (сегментов).

Сегменты задачи могут располагаться в любом месте физической оперативной памяти, защищены друг от друга и от сегментов других задач и вызываются при необходимости их использования в задаче в оперативную память (на свободное место или вместо каких-либо других сегментов этой или других задач) операционной системой.

Виртуальные адреса, используемые в программе задачи, состоят из виртуального номера сегмента и смещения (номера ячейки) в этом сегменте (рис 13.). Виртуальный сегментный адрес аппаратно преобразуется в физический адрес при непосредственном обращении в оперативную память с использованием таблицы соответствия виртуальных номеров сегментов задачи начальным физическим адресам их расположения в оперативной памяти. К такому физическому адресу, соответствующему номеру сегмента в виртуальном адресе, прибавляется величина смещения, указанная в виртуальном адресе. Полученный физический адрес используется для непосредственного обращения к оперативной памяти.

В указанной таблице соответствия помещено также значение длины сегмента, используемого в виртуальном адресе . Указание смещения в сегменте, большего этого значения не допускается. Таким образом обеспечивается и защита сегментов друг от друга.

Существенным преимуществом сегментной организации виртуальной памяти является возможность уникальной защиты помещаемого в сегмент программного объекта по способу обращения к нему. Так, например, модуль процедуры должен быть защищен от возможности осуществления записи в него; запись в модуль данных может быть либо запрещена, либо разрешена. Информация об уникальной защите сегмента также размещается в таблице соответствия. Там же размещается и признак наличия виртуального сегмента в оперативной памяти.

Линейную таблицу соответствия виртуальных номеров сегментов задачи и их мест расположения в физической оперативной памяти (длина такой таблицы соответствует количеству виртуальных сегментов задачи) создает операционная система ЭВМ, поскольку именно она размещает требуемые сегменты в оперативной памяти, передавая их в нее из внешней памяти (выполняется также обратная перепись во внешнюю память сегментов, содержимое которых изменилось, в случае, если их место расположения отведено для размещения новых сегментов. Размещение таблицы соответствия в оперативной памяти требует при замене виртуального адреса на физический одного дополнительного обращения к оперативной памяти, то есть 100% накладных расходов. Существенно избежать этих накладных расходов удается за счет размещения информации из таблицы соответствия в быстродействующих регистрах, в каждом из которых находится как виртуальный номер сегмента, так и алрес его размещения в оперативной памяти. Ассоциативный поиск адреса размещения виртуального сегмента в оперативной памяти происходит в этих регистрах по его виртуальному номеру практически без задержки. Свойство “локальности” программы позволяет почти всегда успешно выполнять такой поиск и осуществлять автоматическую замену виртуального адреса на физический при относительно небольшом числе регистров, содержащих информацию из таблицы соответствия. Редкий “промах” приводит к переписи нужной строки из таблицы соответствия, находящейся в оперативной памяти, в регистр “быстродействующей” таблицы. Выбор номера регистра производится аппаратно.

Значительным недостатком организации виртуальной памяти, разделяемой на сегменты разной длины, является наличие “внешней фрагментации” оперативной памяти, то есть незанятых участков памяти между сегментами. Этот недостаток проявляется при

необходимости расположить в памяти новый сегмент задачи и отсутствии свободного фрагмента оперативной памяти для его размещения. В этом случае приходится принимать меры к освобождению для нового виртуального сегмента участка памяти необходимой длины за счет возможной подвижки в оперативной памяти других виртуальных сегментов или переписи информации (если она изменялась) из освобождаемого участка памяти во внешнюю память.

Страничная организация виртуальной памяти

Страничная организация виртуальной памяти задачи исключает упоминавшийся недостаток сегментной организации – внешнюю фрагментацию. Виртуальная память задачи разделяется на виртуальные “страницы” одинаковой длины, которые могут помещаться на любые физические “страницы” (области оперативной памяти той же длины, что и у виртуальных страниц задачи). Начальные адреса физических страниц имеют значения, кратные длине страницы. Часто используется величина, называемая номером физической страницы и являющаяся значением старших разрядов таких адресов. Количество младших разрядов определяется длиной страницы (младшие разряды, естественно, используются для указания смещения данных в странице).

Аналогично адресам по физической памяти также структурируются виртуальные адреса. Старшие разряды виртуального адреса являются номером виртуальной страницы, а младшие разряды указывают смещение данных в странице. Смещение данных в виртуальной и физической странице одинаковое. Объем виртуальной памяти задачи может не совпадать с объемом физической памяти машины (как правило, превосходит его) и, соответственно, может не совпадать длина виртуальных и физических адресов. При этом разным является лишь количество разрядов, отводимых под номер страницы.

Замена виртуального адреса на физический адрес в целом происходит аналогично такой замене для виртуальной памяти сегментной организации. Используется создаваемая операционной системой таблица виртуальных страниц, каждая строка которой соответствует одной виртуальной странице и содержит номер физической страницы, содержащей информацию этой виртуальной страницы. В этой же строке таблицы находится признак наличия данной виртуальной страницы в физической памяти. Если необходимая виртуальная страница присутствует в физической памяти, то физический адрес для непосредственного обращения в память образуется заменой номера виртуальной страницы на номер физической страницы из соответствующей строки таблицы соответствия. Если же необходимая виртуальная страница в физической памяти отсутствует, то операционная система перемещает ее (“подкачивает”) из внешней памяти на физическую страницу, номер которой определяется с использованием применяемого алгоритма “замещения” страниц в физической памяти. Могут использоваться разные алгоритмы: “первым пришел – первым вышел”, по давности использования и др.

При этом в замещаемой странице может находиться информация как выполняемой задачи, так и других задач. Если содержимое страницы, выбранной для замены, изменялось с момента переписи в нее информации виртуальной страницы из некоторого места внешней памяти, то обратная перепись измененного содержимого страницы в то же место внешней памяти выполняется до осуществления переписи на выбранную физическую страницу информации требуемой виртуальной страницы из внешней памяти.

Использование для замещения страниц таблицы виртуальных страниц в оперативной памяти приводит (как и для сегментной виртуальной памяти) к 100% накладных расходов. Свойство “локальности” программы также как и для случая сегментной виртуальной памяти позволяет успешно использовать для замены адресов “быструю” таблицу “адресной трансляции” на регистрах. Строки такой таблицы содержат как виртуальный номер страницы, так и соответствующий ему номер физической страницы. Ассоциативный поиск для замены виртуального адреса на физический происходит в таблице адресной трансляции по номеру виртуальной страницы. В случае “промаха” при поиске в таблице адресной трансляции

производится перепись необходимой строки в таблицу адресной трансляции из таблицы виртуальных страниц в оперативной памяти.

Несомненным преимуществом страничной виртуальной памяти является обеспечение плотного заполнения физической памяти и эффективной замены информации (страниц) в ней.

При заметном числе страниц, используемых задачей, недоиспользование объема последней страницы (“внутренняя фрагментация”) не является существенным.

Недостатком страничной организации виртуальной памяти является отсутствие имеющейся в сегментной организации виртуальной памяти возможности уникальной защиты объектов программы (программных модулей и модулей данных) в связи с тем, что несколько объектов, требующих разного типа защиты, могут быть помещены в одной странице.

Сегментно-страничная организация виртуальной памяти.

Организация такой памяти преследует цель освободиться от недостатков как сегментной, так и страничной виртуальной памяти (“внешней фрагментации” и отсутствия уникальной защиты объектов программы соответственно).

В сегментно-страничной организации памяти уникально защищаемый виртуальный сегмент разбивается на виртуальные страницы, располагающиеся на любых физических страницах оперативной памяти. Виртуальный сегментно-страничный адрес состоит из трех полей: поля виртуального номера сегмента, поля виртуального номера страницы в этом сегменте, поля смещения в странице. Замена виртуального сегментно-страничного адреса на физический адрес производится так же, как и для сегментного и страничного виртуальных адресов в основном с использованием “быстрой” таблицы адресной трансляции на регистрах. Ассоциативный поиск по этой таблице производится в этом случае по совокупности значений виртуальных номеров сегмента и страницы в нем, находящихся в строках этой таблицы. При совпадении (получающемся в большинстве случаев, благодаря свойству локальности программы) из найденной строки таблицы берется находящийся там физический номер страницы, к которому прибавляется (“присоединяется”) смещение в странице. Полученный физический адрес используется для непосредственного обращения в оперативную память машины.

При “промахе” в поиске по таблице адресной трансляции происходит двухуровневый поиск по таблицам, подготовленным операционной системой в оперативной памяти. Вначале исследуется таблица виртуальных сегментов задачи. Найденная по виртуальному номеру сегмента строка этой таблицы содержит сведения об уникальной защите сегмента, содержащего объект программы, и ссылку на таблицу виртуальных страниц этого сегмента. Найденная по виртуальному номеру страницы строка этой таблицы содержит признак нахождения виртуальной страницы в оперативной памяти и номер физической страницы, в которой находится информация виртуальной страницы.

Указанным выше образом образуется физический адрес для непосредственного обращения в оперативную память и происходит размещение найденной информации в строке “быстрой” таблицы аппаратной трансляции.

При большом количестве виртуальных сегментов, размер которых существенно меньше размеров страницы (для сегмента выделяется целая страница), указанная выше “внутренняя фрагментация” становится заметно отрицательным фактором.

Параллелизм использования внешних устройств ЭВМ

Параллельная обработка информации в вычислительных системах требует в большинстве случаев организации параллельного выполнения многих потоков входных и выходных данных и временного помещения во внешнюю память и считывания из нее промежуточных результатов обработки информации.

Используются два основных способа организации параллельной работы внешних устройств ЭВМ:

- с использованием аппаратуры “селекторных” и “мультиплексных” каналов;
- с использованием аппаратуры “общей шины”.

Селекторные и мультиплексные каналы

Устройства “селекторный канал” и “мультиплексный канал” автоматически осуществляют задаваемые операционной системой (“драйверами” - программами управления работой устройств) обмены данными между оперативной памятью и устройствами, подключенными к этим каналам.

К селекторным каналам через специальную аппаратуру непосредственного управления устройствами (“контроллеры” – control unit) подключаются устройства, обмен данными с которыми производится в режиме монопольного захвата канала. Операционная система выбирает одно из устройств, подключенных к селекторному каналу, и запускает обмен информацией с ним в монопольном режиме использования селекторного канала. Эффективное использование селекторного канала возможно для обмена через него с так называемыми “быстрыми” внешними устройствами. Основными такими устройствами являются устройства внешней памяти на магнитных дисках. К селекторному каналу может быть подключено несколько групп устройств через свое для каждой группы устройство управления работой устройств. К оперативной памяти может быть подключено несколько параллельно работающих селекторных каналов. Осуществлению такой возможности способствует рассмотренное выше “расслоение” оперативной памяти на параллельно работающие блоки.

Управление обменом данными осуществляет устройство “селекторный канал” с использованием подготовленного драйвером операционной системы “управляющего слова” обмена, помещаемого на регистр управляющего слова обмена устройства “селекторный канал”.

Управляющее слово обмена содержит адрес обмениваемых данных по оперативной памяти, постоянно изменяющийся устройством “селекторный канал” в процессе обмена, и, соответственно, постоянно уменьшающуюся в процессе обмена величину количества данных, которые осталось передать (или конечный адрес по оперативной памяти передаваемого массива данных), для контроля окончания обмена. Управляющее слово обмена может быть помещено на регистр управляющего слова в устройстве “селекторный канал” драйвером внешнего устройства непосредственно, или оно может быть взято устройством “селекторный канал” из подготовленной драйвером в оперативной памяти последовательности управляющих слов обмена, содержащих информацию о ряде последовательных обменов с этим внешним устройством (“программа подканала”). Непосредственная работа с внешним устройством (запись/считывание) производится его контроллером по информации, переданной ему драйвером внешнего устройства при подаче команды “пуск обмена”.

По сигналу о готовности конкретного внешнего устройства к выполнению обмена, поступившему из контроллера этого устройства в устройство “селекторный канал” происходит выбор соответствующего номеру внешнего устройства (“подканала”) адресного слова подканала из группы подготовленных драйверами внешних устройств адресных слов подканалов, содержащих “текущие” адреса управляющих слов обмена с внешними устройствами. Выбранное адресное слово подканала передается на регистр адресного слова в устройстве “селекторный канал”. Затем по адресу, размещенному в адресном слове подканала выбирается текущее управляющее слово обмена из программы подканала и помещается на регистр управляющего слова обмена. После этого монопольно происходит описанный выше процесс обмена данными между внешним устройством и оперативной памятью по селекторному каналу. После окончания обмена следующий обмен данными может выполняться с этим же или другим внешним устройством, подключенным к данному селекторному каналу.

Устройство “мультиплексный канал” предназначено для “параллельного” выполнения обменов с подключенными к нему несколькими внешними устройствами. Выполнение таких

обменов производится группами (“блоками”) данных, на которые автоматически разбивается массив данных, передаваемый в каждом обмене. Через мультиплексный канал чередуется (“мультиплексируется”) передача блоков данных (возможно, состоящих даже из одного байта – “байт-мультиплексный” режим) по разным “подканалам” (внешним устройствам), подключенным к мультиплексному каналу. Цикл работы устройства “мультиплексный канал” похож на описанный выше цикл работы устройства “селекторный канал”. Разница заключается в том, что по сигналу внешнего устройства о готовности к передаче данных через мультиплексный канал осуществляется передача лишь “текущего” блока данных из всего обмениваемого массива данных, после чего измененное состояние управляющего слова обмена (по адресу оперативной памяти и количеству еще не переданных данных) переписывается из регистра управляющего слова обмена устройства “мультиплексный канал” обратно в программу подканала.

Обмен блоками данных по мультиплексному каналу производится между оперативной памятью и выделенным для внешнего устройства буфером в памяти его контроллера. Прием данных из внешнего устройства в этот буфер или выдача данных из этого буфера во внешнее устройство происходит под управлением контроллера устройства. Рассмотренная организация работы мультиплексного канала позволяет обеспечить фактически параллельную работу нескольких подключенных к нему «медленных» внешних устройств (например, терминалов ввода-вывода информации).

В устройствах «селекторный канал» и «мультиплексный канал» после окончания обмена с внешним устройством изменяется адресное слово подканала, чтобы можно было по находящемуся в нем адресу найти в программе подканала новое управляющее слово обмена с данным внешним устройством.

Многомашинные вычислительные системы

Многомашинные вычислительные системы стали возникать практически сразу после появления первых ЭВМ. Появление новой машины (как правило, более производительной) в какой-либо организации стимулировало объединение машин в основном в целях организации “разделения труда” – менее производительной ЭВМ поручалась организация ввода-вывода данных, а более производительная ЭВМ выполняла основную обработку информации. Более развитым вариантом “разделения труда” в многомашинном комплексе является организация “конвейера ЭВМ”, в котором каждая ЭВМ выполняет свой этап обработки поступающих порций входной информации и передает полученные результаты другой ЭВМ для дальнейшей обработки. Достигаемая при этом высокая производительность существенна для эффективной работы систем обработки информации в реальном времени. ЭВМ объединялись и в целях резервирования (в том числе “горячего” резервирования - для срочного продолжения обработки информации при отказе ЭВМ), а также в целях контроля правильности вычислений - выполнялась одна и та же обработка информации на разных ЭВМ и сравнение (в том числе поэтапное) полученных результатов. Имеющие в настоящее время широкое распространение “локальные” и “глобальные” сети ЭВМ являются логическими подмножествами класса многомашинных комплексов.

Естественным является классификационное разделение класса многомашинных вычислительных систем на системы с “сильной” и “слабой” связью входящих в них ЭВМ. К системам с “сильной” связью относятся системы, в которых передача данных от одной машины к другой происходит со скоростью порядка скорости работы оперативной памяти машины. Это может быть реализовано при использовании быстрых каналов, связывающих оперативные памяти машин (например, рассмотренных выше селекторных каналов), или при использовании общедоступной для всех машин оперативной памяти. В системах с “слабой” связью данные могут передаваться между машинами по медленным каналам (например, телефонным) или через промежуточное помещение во внешнюю память, доступную для машин, входящих в систему (например, на магнитные диски через контроллер, имеющий несколько входов).

Многомашинный вычислительный комплекс с общедоступной памятью

Построение такого комплекса преследовало несколько целей: организацию работы упомянутого выше “конвейера ЭВМ”, использование общих внешних устройств (к внешним устройствам, подключенным к одной ЭВМ, могли иметь доступ задачи, решаемые на других ЭВМ), резервирование.

В систему входили универсальные ЭВМ, отдельные устройства оперативной памяти, специализированные ЭВМ управления работой внешних устройств. Эти основные компоненты соединялись с помощью быстродействующей сети с коммутаторами, содержащими информацию о структуре (“картине”) сети. По такой сети передавались одиночные “сообщения”, содержащие передаваемые данные и адреса их “источников” и “приемников” (ячеек оперативной памяти, регистров процессоров). По адресной информации в сообщениях коммутаторы, имея “картину” сети, передавали их по нужным направлениям сети. Передача групп данных осуществлялась с помощью разбиения их на отдельные передаваемые сообщения.

К специализированным (“периферийным”) машинам подключался набор внешних устройств, управляемых операционными системами этих машин. Заявки на выполнение обменов с внешними устройствами передавались этим операционным системам операционными системами других машин комплекса.

Фактически обеспечивалась параллельная работа (в том числе в режиме “конвейера”) всех ЭВМ комплекса.

СуперЭВМ как многомашинный вычислительный комплекс

Большой объем обработки информации, осуществляемый в суперЭВМ, поддерживается развитыми средствами управления обработкой, подготовки заданий на обработку, подготовки и передачи исходных данных, запоминания и выдачи результатов на устройства вывода информации. Этими средствами являются так называемые “внешние” ЭВМ (“машины-спутники”, “front-end computers”), подключаемые через каналы различного типа к “основному вычислителю” (многопроцессорному комплексу или кластеру).

Эффективное использование большой производительной суперЭВМ во многих случаях требует организации развитой телекоммуникационной связи с ней многих удаленных абонентов для загрузки суперЭВМ заданиями на обработку данных.