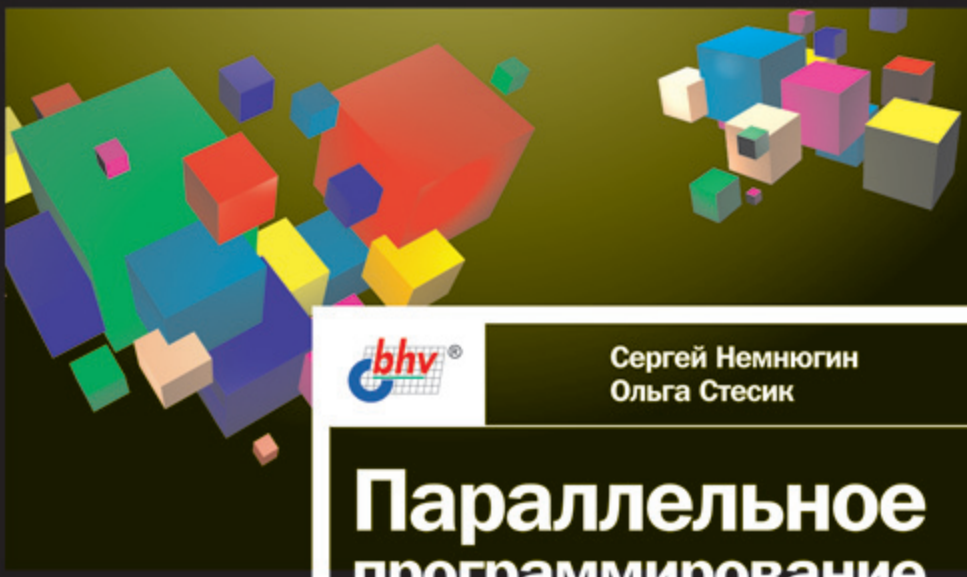


Сергей Немнюгин
Ольга Стесик

Параллельное программирование для многопроцессорных вычислительных систем

- Архитектура высокопроизводительных вычислительных систем
- Модели и методы параллельного программирования
- Разработка и отладка параллельных программ

МАСТЕР_ПРОГРАММ_

Сергей Немнюгин
Ольга Стесик

Параллельное программирование для многопроцессорных вычислительных систем

Санкт-Петербург
«БХВ-Петербург»
2002

УДК 681.3.06
ББК 32.973.26-018.1
Н50

Немнюгин С. А., Стесик О. Л.

Н50 Параллельное программирование для многопроцессорных вычислительных систем. — СПб.: БХВ-Петербург, 2002. — 400 с.: ил.

ISBN 5-94157-188-7

Книга является практическим руководством для разработки прикладного программного обеспечения параллельных многопроцессорных систем. Приводятся сведения об архитектуре высокопроизводительных систем параллельного программирования — MPI (Message Passing Interface), PVM (Parallel Virtual Machine), HPF (High Performance Fortran). Излагается методика параллельного программирования для создания своих эффективных параллельных (и векторизованных) программ. Представленные примеры помогут разобраться в тонкостях работы многопроцессорных систем, а задания-упражнения для самостоятельной работы — закрепить изложенный материал. В приложениях описываются способы отладки параллельных кластеров, методы исследования производительности, дан обзор средств визуализации исполнения многопроцессорных приложений.

Для программистов, преподавателей и студентов, чья деятельность связана с высокопроизводительными вычислениями

УДК 681.3.06
ББК 32.973.26-018.1

Группа подготовки издания:

Главный редактор	*Екатерина Кондукова*
Зам. главного редактора	*Анатолий Адаменко*
Зав. редакцией	*Анна Кузьмина*
Редактор	*Вячеслав Каштанов*
Компьютерная верстка	*Ольги Сергиенко*
Корректор	*Зинаида Дмитриева*
Оформление серии	*Via Design*
Дизайн обложки	*Игоря Цырульникова*
Зав. производством	*Николай Тверских*

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 05.09.02.
Формат 70×100 1/16. Печать офсетная. Усл. печ. л. 32,25.
Тираж 3000 экз. Заказ №
"БХВ-Петербург", 198005, Санкт-Петербург, Измайловский пр., 29.

Гигиеническое заключение на продукцию, товар № 77.99.02.953.Д.001537.03.02
от 13.03.2002 г. выдано Департаментом ГСЭН Минздрава России.

Отпечатано с готовых диапозитивов
в Академической типографии "Наука" РАН
199034, Санкт-Петербург, 9 линия, 12.

ISBN 5-94157-188-7

Содержание

Предисловие

Идея параллельной обработки данных не нова. Можно считать, что она возникла еще на заре человеческой цивилизации, когда оказалось, что племя может успешно бороться за выживание, если каждый его член выполняет свою часть общей работы. А представьте себе учреждение, где один человек работает в качестве директора, секретаря директора, курьера, успевает поработать у станка, отремонтировать электропроводку, а ночью еще и охраняет свой объект. Выглядит это почти нелепо. Можно допустить, что такая организация выполнит определенную работу, но времени на это уйдет очень много. А ведь в таком режиме, фактически, работает обычный последовательный компьютер, справляясь с решением поставленных перед ним задач только благодаря быстрой работе центрального процессора. Совсем другое дело, когда в организации есть несколько сотрудников, которые работают одновременно, и каждый из них решает свою часть общей задачи. Можно считать, что режим параллельной обработки данных является наиболее естественным "режимом" существования и человеческого общества.

Неудивительно, что еще на заре компьютерной эры, примерно в середине прошлого века, конструкторы электронно-вычислительных машин задумались над возможностью применения параллельных вычислений в компьютерах. Ведь увеличение быстродействия только за счет совершенствования электронных компонентов компьютера — достаточно дорогой способ, который, к тому же, сталкивается с ограничениями, налагаемыми физическими законами. Так параллельная обработка данных и параллелизм команд были введены в конструкцию компьютеров и сейчас любой пользователь "персоналки", возможно, сам того не зная, работает на параллельном компьютере. Впрочем, обычному пользователю персонального компьютера и не надо об этом знать.

Вместе с тем, достаточно часто приходится сталкиваться с такими задачами, которые, представляя немалую ценность для общества, не могут быть решены с помощью относительно медленных компьютеров офисного или до-

машнего класса. Единственная надежда в этом случае возлагается на компьютеры с большим быстродействием, которые принято называть суперкомпьютерами. Только машины такого класса могут справиться с обработкой больших объемов информации. Это могут быть, например, статистические данные или результаты метеорологических наблюдений, финансовая информация. Иногда скорость обработки имеет решающее значение. В качестве примера можно привести составление прогноза погоды и моделирование климатических изменений. Чем раньше предсказано стихийное бедствие, тем больше возможностей подготовиться к нему. Важной задачей является моделирование лекарственных средств, расшифровка генома человека. А томография, в том числе и медицинская? А разведка месторождений нефти и газа? Примеров можно привести много.

В середине 60-х годов прошлого века появились первые суперкомпьютеры, это были параллельный 64-процессорный компьютер ILLIAC-IV и векторный компьютер CDC 6500 (рис. 1). Последний стал дебютом легендарного конструктора высокопроизводительных вычислительных систем Сеймура Крея (рис. 2).

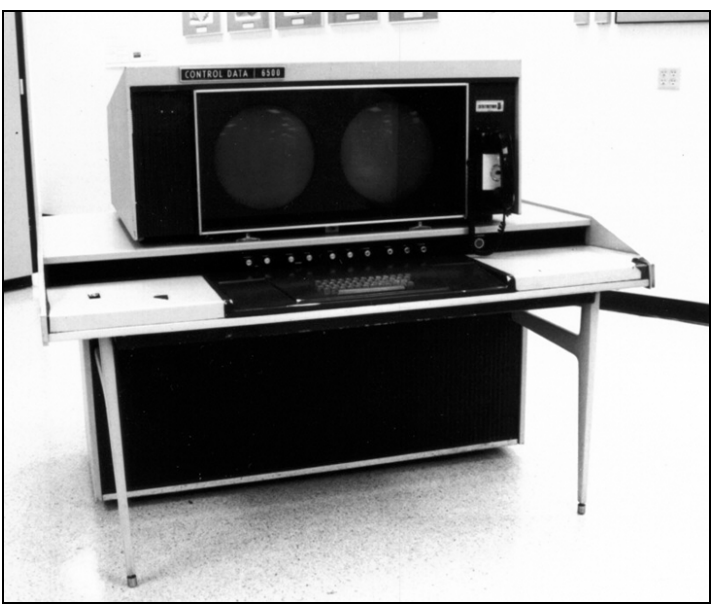


Рис. 1. Компьютер CDC 6500

Сеймур Крей основал знаменитую фирму Cray, которая и сейчас выпускает одноименные суперкомпьютеры (рис. 3 и 4).

Из современных проектов следует упомянуть ASCI (Accelerated Strategic Computer Initiative) — проект, в рамках которого создаются вычислительные

системы с рекордной производительностью. В качестве примера можно привести суперкомпьютеры ASCI Red и ASCI White, которые вошли в рейтинг наиболее мощных вычислительных систем мира "Top 500 Supercomputers". Правительство США вкладывает в этот проект большие деньги, поскольку одним из применений таких систем является имитационное моделирование ядерного оружия, которое рассматривается в качестве альтернативы реальным испытаниям.

Вспоминая историю развития суперкомпьютерных технологий, нельзя не сказать о вкладе тогда еще советских ученых и конструкторов. Авторам довелось поработать на таких машинах высочайшего класса, как БЭСМ-6 и "Эльбрус".

Рис. 2. Сеймур Крей

Рис. 3. Суперкомпьютер Cray-1, теперь уже история

Рис. 4. Суперкомпьютер SV1, день сегодняшний

Суперкомпьютеры являются дорогими, элитными машинами, стоимость которых может достигать десятков миллионов долларов. Одним из решений проблемы доступа к дорогостоящим суперкомпьютерным ресурсам является создание центров, обслуживающих несколько организаций. Так, в США имеется около десятка крупных суперкомпьютерных центров. Имеются такие центры и в России, наиболее крупные из них расположены в Москве и Санкт-Петербурге.

Примерно в середине 80-х годов прошлого века ученые заинтересовались возможностью использования кластерных систем в качестве относительно недорогих заменителей суперкомпьютеров. Благодаря развитию сетевых технологий, появилась возможность соединять вместе несколько рабочих станций, в качестве которых могут использоваться обычные персональные компьютеры. Ну а чем такая система не многопроцессорный вычислительный комплекс? Было разработано необходимое программное обеспечение, которое стало распространяться бесплатно. Работало это программное обеспечение в среде операционной системы Linux, которая тоже является бесплатной, оставаясь в то же время чрезвычайно гибкой и надежной. Таким образом, основная часть программного обеспечения кластера оказывается бесплатной, что значительно снижает суммарную стоимость всей системы. Оказалось, что в результате можно построить многопроцессорный комплекс, который лишь в 2—3 раза уступает по быстродействию суперкомпьютеру, но дешевле его в несколько сотен раз. Такие системы, которые получили название Linux-ферм, быстро завоевали популярность. На рис. 5 изображен 40-процессорный кластер, установленный в Санкт-Петербургском государственном университете (Петродворцовый телекоммуникационный центр). Этот кластер активно используют ученые университета, физики, химики, математики для решения сложнейших научных задач.

Рис. 5. Суперкластер Санкт-Петербургского государственного университета

Следует отметить, что хотя исследования в области применения кластеров на Западе ведутся уже достаточно давно, в России интерес к ним появился сравнительно недавно. Отрадно, что "суперкомпьютинг" получил реальную поддержку со стороны государства. Ведь суперкомпьютерные ресурсы недаром относят к числу стратегических. Принята государственная программа развития суперкомпьютерных технологий, созданы суперкомпьютерные центры, ведется обучение специалистов. К сожалению, ощущается недостаток учебной литературы, посвященной вопросам параллельного программирования. Изданы книги по общим вопросам архитектуры параллельных и векторных вычислительных систем, по параллельным алгоритмам, а вот описания средств разработки параллельных программ нет... Авторы надеются, что их скромные усилия позволят восполнить этот недостаток. Наша книга рассчитана на программиста, уже имеющего определенный опыт разработки обычных, последовательных программ.

Немного о структуре книги. В *главе 1* рассматривается архитектура высокопроизводительных ЭВМ. Здесь мы знакомим читателя с основными концепциями архитектуры высокопроизводительных вычислительных систем, такими как конвейеры данных и команд, векторная обработка данных. Рассматриваются особенности устройства памяти, подсистемы коммуникаций. Общее понимание принципов работы высокопроизводительной вычислительной системы полезно разработчику параллельных программ.

В *главе 2* речь идет об особенностях программирования параллельных и векторных вычислений. Здесь мы уделим внимание технологии разработки параллельных алгоритмов, познакомимся с различными моделями программирования. Эта глава завершает часть книги, посвященную общим вопросам параллельного программирования.

Остальная часть содержит описание наиболее распространенных программных средств разработки параллельных программ. *Глава 3* служит введением в одну из реализаций спецификации MPI (Message Passing Interface) — MPICH, которая основана на модели передачи сообщений. Рассматривают-

ся общая организация и структура MPICH, организация обмена данными. Вводится концептуальное для MPI понятие коммуникатора. Описывается структура MPI-программы, компиляция и запуск MPI-программ.

В *главе 4* излагаются средства MPICH, с помощью которых организуется обмен сообщениями. Рассматриваются различные режимы двухточечного обмена, коллективный обмен, управление группами процессов. Даются описания подпрограмм библиотеки MPICH, приводятся примеры программ.

В *главе 5* рассматриваются более сложные вопросы программирования с использованием MPI. Это конструирование производных типов данных, топологии и другие вопросы.

В *главе 6* даются основы другой популярной системы параллельного программирования — PVM (Parallel Virtual Machine). Рассматривается архитектура PVM, настройка системы, работа с PVM-консолью. Дано описание основных подпрограмм системы. Приведен сравнительный обзор MPI и PVM, который позволит читателю при необходимости выбрать оптимальную для его задач систему параллельного программирования.

Глава 7 содержит описание подпрограмм библиотек PVM для обмена сообщениями, управления процессами и виртуальной машиной. Разбираются такие вопросы, как динамическое изменение конфигурации виртуальной машины, буферизация данных и некоторые другие.

В последней, *главе 8* дается описание языка Высокопроизводительный FORTRAN, а также необходимые для понимания материала сведения о языке FORTRAN-90. Приводятся примеры программ.

В *приложениях* читатель найдет описание средств отладки и мониторинга параллельных программ, рекомендации по созданию и наладке своего собственного параллельного кластера и краткий обзор ресурсов Интернета, посвященных высокопроизводительным вычислениям.

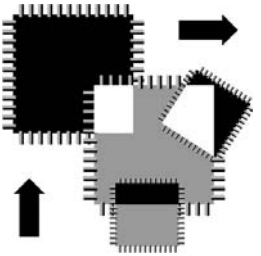
В книге приведены многочисленные примеры параллельных программ. В конце каждой главы приводятся вопросы и задания для самостоятельной работы. Мы рекомендуем не пренебрегать выполнением этих заданий. Полезен и самостоятельный подробный анализ примеров.

Мы будем признательны за замечания и предложения по содержанию книги, которые можно направлять по адресу электронной почты:

nemnugin@mph.phys.spbu.ru

Авторы с удовольствием выражают благодарность директору Петродворцового телекоммуникационного центра СПбГУ В. И. Золотареву, директору Междисциплинарного центра СПбГУ Н. В. Борисову, поддержка которых в немалой степени способствовала появлению книги. Благодарим доцента физического факультета А. В. Комолкина, своих студентов и коллег. Мы признательны редакции издательства "БХВ-Петербург" за поддержку проекта.

ГЛАВА 1



Архитектура высокопроизводительных ЭВМ

Прежде чем мы перейдем к изучению методов и средств параллельного программирования, полезно познакомиться с некоторыми особенностями устройства высокопроизводительных вычислительных систем. Преимуществом использования языков программирования высокого уровня является универсальность программ и их простая переносимость между различными компьютерами (разумеется, если на этих компьютерах имеется необходимое программное обеспечение, прежде всего, трансляторы). Следует учитывать, что эффективность выполнения параллельных программ определяется не только аппаратной частью, но и способностью транслятора генерировать эффективный исполняемый код. Оба эти фактора взаимосвязаны и порой сложно определить, какой из них имеет решающее значение.

Обычно, программист полагается на эффективность транслятора, считая, что при грамотном программировании сгенерированный транслятором исполняемый код будет обладать хорошими показателями по быстродействию; в этом случае отпадает необходимость в применении языков низкого уровня (ассемблеров), требующих глубоких знаний архитектуры процессора.

Но, бывает и так, что программисту все же требуется знание устройства компьютера, принципов и даже некоторых деталей его работы. Можно привести следующий пример. При разработке программ для параллельных ЭВМ используются специализированные библиотеки, позволяющие организовать обмен данными между отдельными подзадачами и их синхронизацию. Для эффективной организации обмена данными необходимо знать, как происходит пересылка информации между процессорами, какова ее скорость и т. д.

В вычислительной технике используются три термина, связанные с устройством электронно-вычислительной машины:

1. *Архитектура компьютера* — это описание основных компонентов компьютера и их взаимодействия. Можно считать, что архитектура — это те атрибуты вычислительной системы, которые видны программисту: набор

команд, разрядность машинного слова, механизмы ввода/вывода, методы адресации. Можно дать и другое определение: архитектура — это внутренняя структура системы или микропроцессора, которая определяет их функциональные возможности и быстродействие.

2. *Организация компьютера* — это описание конкретной реализации архитектуры, ее воплощения "в железе". Оно включает перечень используемых сигналов управления, интерфейсов, технологий памяти, реализацию арифметических, логических и других операций. Так, умножение может быть реализовано на аппаратном уровне или посредством многократного повторения операции суммирования. Ряд суперкомпьютеров Cray, например, имеют сходную архитектуру. С точки зрения программиста у них одинаковое число внутренних регистров, используемых для временного хранения данных, одинаковый набор машинных команд, одинаковый формат представления данных. Организация же компьютеров Cray разных моделей может существенно различаться. У них может быть разное число процессоров, разный размер оперативной памяти, разное быстродействие и т. д.
3. *Схема компьютера* — детальное описание его электронных компонентов, их соединений, устройств питания, охлаждения и др. Программисту довольно часто требуется знание архитектуры компьютера, реже — его организации и никогда — схемы компьютера. Действительно, зачем специалисту, который занимается, например, расчетом прочностных свойств новой модели автомобиля, знать, какая марка вентилятора охлаждает процессор компьютера во время выполнения его программы?

Классификация Флинна

Одной из наиболее известных схем классификации компьютерных архитектур является *таксономия Флинна*, предложенная Майклом Флинном в 1972 году. В ее основу положено описание работы компьютера с потоками команд и данных. В классификации Флинна имеется четыре класса архитектур:

1. **SISD** (Single Instruction Stream — Single Data Stream) — один поток команд и один поток данных.
2. **SIMD** (Single Instruction Stream — Multiple Data Stream) — один поток команд и несколько потоков данных.
3. **MISD** (Multiple Instruction Stream — Single Data Stream) — несколько потоков команд и один поток данных.
4. **MIMD** (Multiple Instruction Stream — Multiple Data Stream) — несколько потоков команд и несколько потоков данных.

Рассмотрим эту классификацию более подробно.

SISD-компьютеры

SISD-компьютеры (рис. 1.1) — это обычные последовательные компьютеры, выполняющие в каждый момент времени только одну операцию над одним элементом данных. Большинство современных персональных ЭВМ принадлежат именно этой категории.

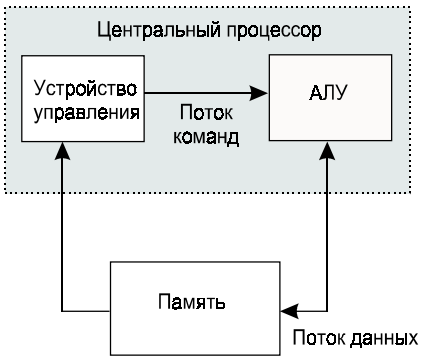


Рис. 1.1. Схема SISD-компьютера

Многие современные вычислительные системы включают в свой состав несколько процессоров, но каждый из них работает со своим независимым потоком данных, относящимся к независимой программе. Такой компьютер является, фактически, набором SISD-машин, работающих с независимыми множествами данных.

SIMD-компьютеры

SIMD-компьютеры (рис. 1.2 и 1.3) состоят из одного командного процессора (управляющего модуля), называемого *контроллером*, и нескольких модулей обработки данных, называемых *процессорными элементами* (ПЭ). Количество модулей обработки данных таких машин может быть от 1024 до 16 384. Управляющий модуль принимает, анализирует и выполняет команды. Если в команде встречаются данные, контроллер рассылает на все ПЭ команду, и эта команда выполняется либо на нескольких, либо на всех процессорных элементах. Процессорные элементы в SIMD-компьютерах имеют относительно простое устройство, они содержат арифметико-логическое устройство (АЛУ), выполняющее команды, поступающие из устройства управления (УУ), несколько регистров и локальную оперативную память. Одним из преимуществ данной архитектуры считается эффективная реализация логики вычислений. До половины логических команд обычного процессора связано с управлением процессом выполнения машинных команд, а остальная их часть относится к работе с внутренней памятью процессора

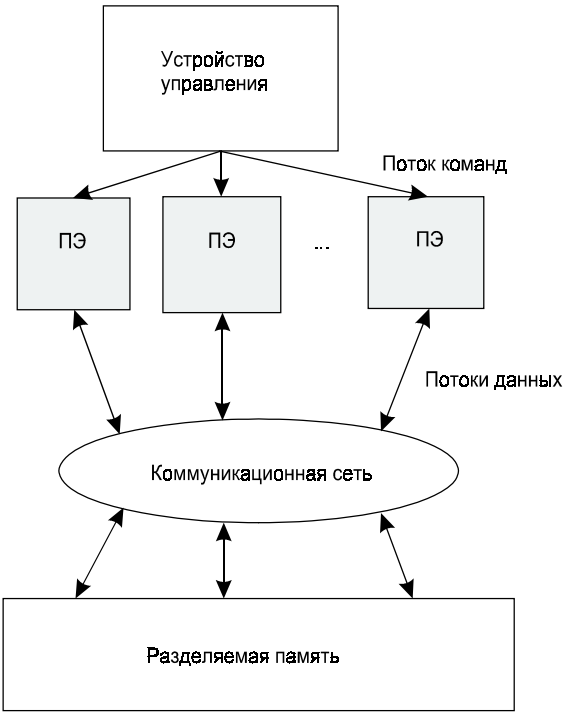


Рис. 1.2. Схема SIMD-компьютера с разделяемой памятью

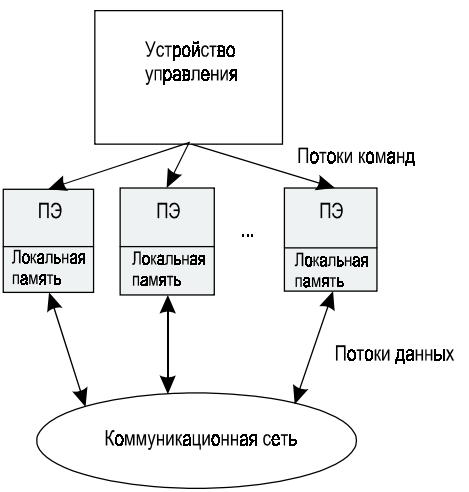


Рис. 1.3. Схема SIMD-компьютера с распределенной памятью

и выполнению арифметических операций. В SIMD-компьютере управление выполняется контроллером, а "арифметика" отдана процессорным элементам. Подклассом SIMD-компьютеров являются векторные компьютеры. Пример такой вычислительной системы — Hitachi S3600.

Другой пример SIMD-компьютера — *матричные процессоры* (Array Processor). В качестве примера можно привести вычислительную систему Thinking Machines CM-2, где 65 536 ПЭ связаны между собой сетью коммуникаций с топологией "гиперкуб". Часто компьютеры с SIMD-архитектурой специализированы для решения конкретных задач, допускающих матричное представление. Это, например, могут быть задачи обработки изображений, где каждый модуль обработки данных работает на получение одного элемента конечного результата.

MISD-компьютеры

Вычислительных машин такого класса мало. Один из немногих примеров — *систолический массив* процессоров, в котором процессоры находятся в узлах регулярной решетки. Роль ребер в ней играют межпроцессорные соединения, все ПЭ управляются общим тактовым генератором. В каждом цикле работы любой ПЭ получает данные от своих соседей, выполняет одну команду и передает результат соседям. На рис. 1.4 дана схема фрагмента систолического массива.

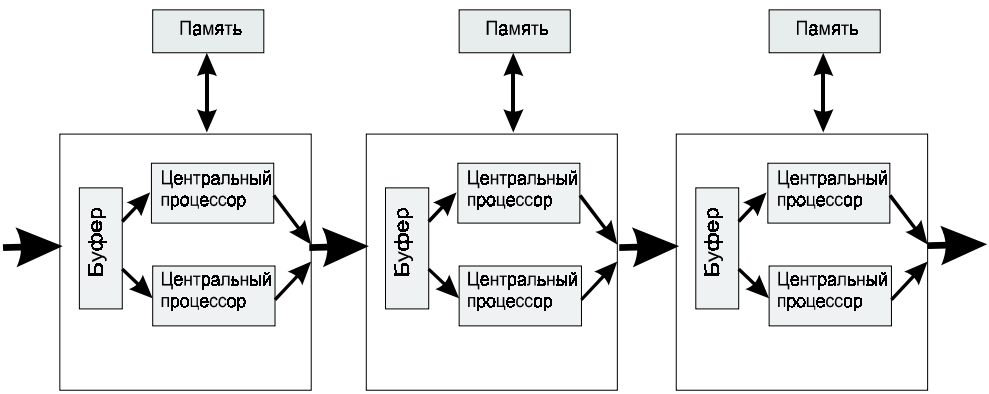


Рис. 1.4. Схема MISD-компьютера

MIMD-компьютеры

Этот класс архитектур (рис. 1.5 и 1.6) наиболее богат примерами успешных реализаций. В него попадают симметричные параллельные вычислительные системы, рабочие станции с несколькими процессорами, кластеры рабочих

станций и т. д. Довольно давно появились компьютеры с несколькими независимыми процессорами, но вначале на них был реализован только принцип параллельного исполнения заданий, т. е. на разных процессорах одновременно выполнялись независимые программы. Разработке первых компьютеров для параллельных вычислений были посвящены проекты под условным названием CM* и C.MMP в университете Карнеги (США). Технической базой для этих проектов были процессоры DEC PDP-11. В начале 90-х годов прошлого века именно MIMD-компьютеры вышли в лидеры на рынке высокопроизводительных вычислительных систем.

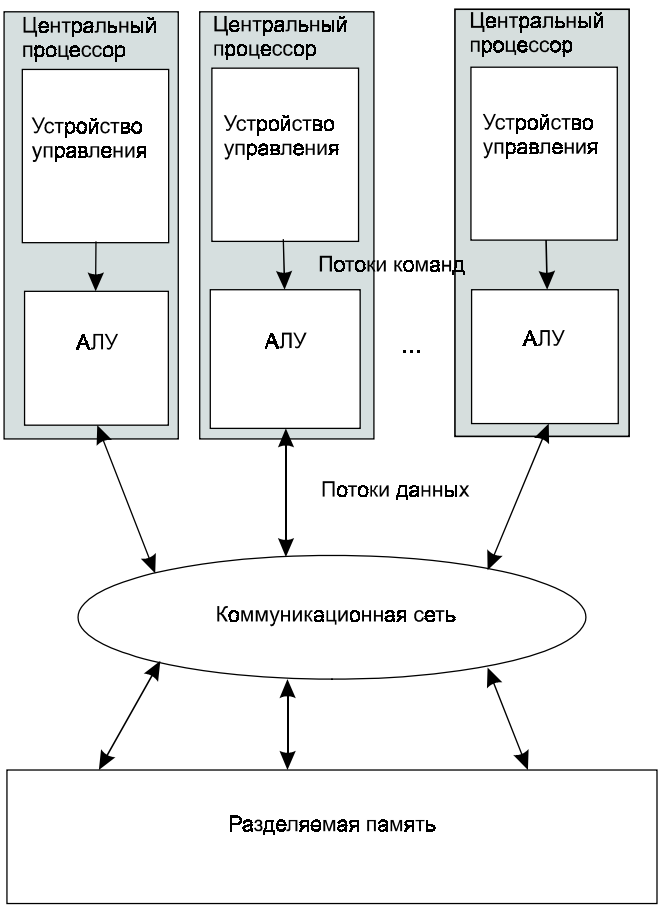


Рис. 1.5. Схема MIMD-компьютера с разделяемой памятью

Имеются и гибридные конфигурации, в которых, например, объединены несколько SIMD-компьютеров, в результате чего получается MSIMD-ком-

пьютер, позволяющий создавать виртуальные конфигурации, каждая из которых работает в SIMD-режиме.

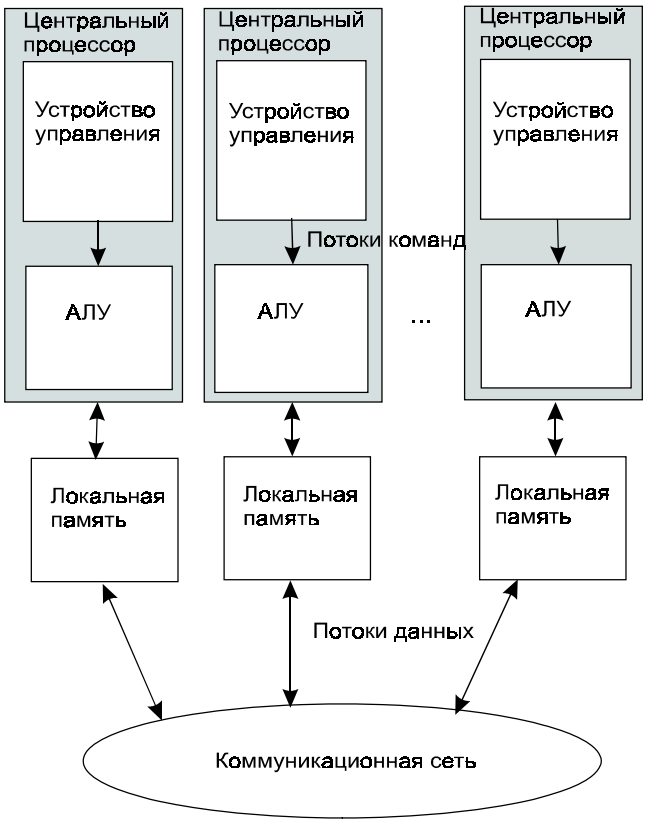


Рис. 1.6. Схема MIMD-компьютера с распределенной памятью

Классификация Флинна не дает исчерпывающего описания разнообразных архитектур MIMD-машин, порой существенно отличающихся друг от друга. Например, существуют такие подклассы MIMD-компьютеров, как системы с разделяемой памятью и системы с распределенной памятью. Системы с разделяемой памятью могут относиться по классификации Флинна как к MIMD, так и к SIMD-машинам. То же самое можно сказать и о системах с распределенной памятью.

Развитием концепции MIMD-архитектуры с распределенной памятью является *распределенная обработка*, когда вместо набора процессоров в одном корпусе используются компьютеры, связанные достаточно быстрой сетью. Концептуального отличия от MIMD-архитектуры с распределенной памятью нет, а особенностью является медленное сетевое соединение.

Традиционная архитектура фон Неймана

Традиционная фон-неймановская архитектура компьютера представлена на рис. 1.7. Она основана на следующих принципах:

- программа хранится в компьютере;
- программа во время выполнения и необходимые для ее работы данные находятся в оперативной (главной) памяти;
- имеется *арифметико-логическое устройство*, выполняющее арифметические и логические операции с данными;
- имеется *устройство управления*, которое интерпретирует команды, выбираемые из памяти, и выполняет их;
- *устройства ввода и вывода* (ВВ) используются для ввода программ и данных и для вывода результатов расчетов. Работают под управлением УУ.

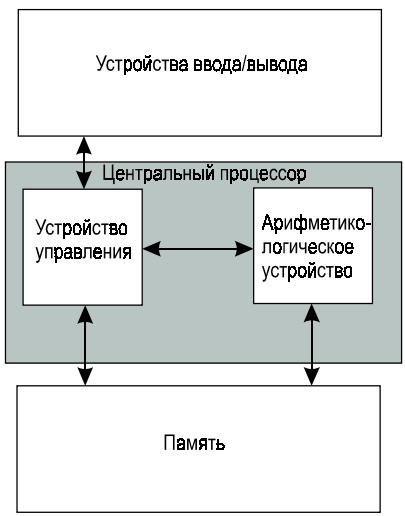


Рис. 1.7. Схема компьютера фон Неймана

Устройство управления и арифметико-логическое устройство (АЛУ) вместе составляют *центральный процессор* (ЦП). Можно считать, что все остальные функциональные блоки компьютера обслуживают АЛУ, а оно выполняет основную работу. Арифметические операции выполняются с целыми и вещественными числами, для представления которых используется двоичный формат. Для отрицательных значений применяется дополнительный код, что позволяет упростить реализацию арифметических и логических операций. Управляющее же устройство управляет работой остальных функциональных узлов компьютера.

В состав "традиционного" фон-неймановского компьютера входит один ЦП, основной задачей которого является выполнение машинных команд, выбираемых из оперативной памяти по очереди, одна за другой. Устройство управления интерпретирует очередную команду, которая входит в набор команд процессора, включающих арифметические и логические операции, операции пересылки данных и некоторые другие. Затем выполняется выборка данных, их обработка и запись результата выполнения в оперативную память. ЦП имеет набор регистров — устройств для временного хранения промежуточных результатов и данных, необходимых для выполнения команд (операндов).

Таким образом, основными компонентами компьютера являются:

- центральный процессор и оперативная память, образующие ядро системы;
- вторичная ("внешняя") память и устройства ввода/вывода, образующие "периферию";
- коммуникации между компонентами системы, осуществляемые посредством шин.

Регистры

Регистры центрального процессора представляют собой вспомогательную память, находящуюся на вершине иерархической организации памяти (об иерархии памяти мы поговорим чуть позже). Количество регистров различается в процессорах разного типа. Программисту доступны:

- регистры общего назначения (General Purpose Registers);
- регистры данных (Data Registers);
- регистры адреса (Address Registers);
- регистр кодов условий (Condition Codes Register).

Имеются также управляющие регистры и регистры состояния:

- счетчик команд (Program Counter);
- регистр декодирования команд (Instruction Register);
- регистр адресации памяти (Memory Addressing Register);
- регистр буфера памяти (Memory Buffer Register).

Выполнение команды

Выполнение команды состоит из двух этапов:

- выборка;
- выполнение.

Первый шаг — выборка — выполняется следующим образом. Вначале из счетчика команд выбирается адрес очередной команды, при этом значение, содержащееся в счетчике команд, увеличивается. Адрес команды помещается в регистр адресации памяти и передается в адресную шину. УУ считывает данные из оперативной памяти. Результат считывания помещается в шину данных, копируется в регистр буфера памяти, а затем в регистр декодирования команд.

При выборке данных анализируется регистр декодирования команд и производится выборка данных. Младшие биты регистра буфера памяти передаются в регистр адресации памяти, затем УУ формирует запрос к памяти и результат выполнения этого запроса (адрес или операнд) пересылается в регистр буфера памяти.

Второй шаг — выполнение. Во время этого цикла происходит пересылка данных между центральным процессором и оперативной памятью и между центральным процессором и модулем ввода/вывода; выполняются арифметические и логические операции над данными. При этом, если есть переходы, может изменяться последовательность выполнения команд.

Набор команд процессора

Центральный процессор выполняет машинные команды из набора, определенного для данной разновидности процессоров. Для представления этих команд используется двоичный формат. Каждая машинная команда включает код операции, ссылку на операнды, адрес записи результата операции и ссылку на следующую по порядку команду.

Набор команд процессора характеризуется:

- типами операндов;
- форматом команд;
- допустимыми командами;
- количеством адресов в команде;
- доступом к регистрам;
- режимами адресации.

Большинство машинных команд можно разделить на четыре категории:

- арифметические и логические операции;
- операции пересылки данных между оперативной памятью и регистрами центрального процессора;
- операции ввода/вывода;
- команды управления (проверки, ветвления).

Пример последовательности машинных команд для вычисления арифметического выражения и оператора присваивания `Z := (X + Y) * 7` приведен в табл. 1.1.

Таблица 1.1. *Пример последовательности машинных команд*

Адрес команды	Команда
00001000	`00001 01110001 111` `Move адрес Y Регистр 7`
00001001	`00011 01110000 111` `Add адрес X Регистр 7`
00001010	`00101 00000011 111` `Mul операнд 3 Регистр 7`
00001011	`00010 01110010 111` `Move адрес Z Регистр 7`

Каждая команда состоит из последовательности элементарных шагов, которые в совокупности составляют цикл команды. Эти шаги являются более "мелкими" составными частями команды, чем упоминавшиеся ранее этапы выборки и выполнения. Управление выполнением команд реализуется как аппаратными средствами, так и средствами микропрограммирования.

Шины

В состав компьютера входят шины. *Шина* — это коммуникационная линия, соединяющая несколько (два или более) устройств. *Шины данных* используются для передачи данных. От ширины или разрядности шины зависит ее производительность, т. е. скорость передачи информации. *Адресная шина* определяет источник и приемник данных. *Шина управления* передает управляющую информацию — сигналы считывания и записи в память, запросы на прерывания, блокирующие сигналы, обеспечивает синхронизацию устройств. При подключении к одной шине нескольких устройств возникают задержки с передачей данных, поскольку в каждый момент времени воспользоваться шиной может только одно устройство, все остальные вынуждены в это время простаивать. Избавиться от таких задержек можно, применяя несколько шин, т. е. распараллеливая процесс передачи информации.

Память

Основными компонентами подсистемы памяти компьютера являются:

- *оперативная (главная) память* — характеризуется высокой скоростью работы (малым временем выборки), сравнительно небольшим объемом и повышенной стоимостью;

- *внешняя (вторичная) память* — имеет большой объем, она дешевле, но характеризуется меньшей скоростью доступа.

В оперативной памяти компьютера размещается программа во время выполнения. Внешняя память используется для долговременного, постоянного хранения информации, в том числе программ, данных и т. д. В качестве устройств длительного хранения информации применяются жесткие диски, RAID-массивы, диски CD-ROM, магнитные ленты, дискеты и другие носители информации. Для выполнения программа загружается в оперативную память из внешней памяти. Важнейшими характеристиками памяти являются ее объем и скорость (время) доступа.

Оперативную память можно представлять себе состоящей из ячеек памяти — минимальных элементов, к которым можно обратиться из программы. Каждая ячейка имеет свой уникальный адрес. Имеются также буфер адресов (содержит адреса ячеек памяти, из которых производится считывание информации, или в которые производится запись), буфер данных (содержит данные для записи в ячейку памяти или для считывания из ячейки), а также декодер адреса и устройство управления памятью. Информация, находящаяся в оперативной памяти, пропадает при выключении компьютера, а информация, содержащаяся во внешней памяти, сохраняется и после выключения питания.

Для эффективной работы компьютера необходимо, чтобы операции с данными в оперативной памяти выполнялись со скоростью, сравнимой со скоростью работы центрального процессора. С другой стороны, известно, что скорость работы с памятью тем меньше, чем больше ее объем. Быстродействующая память сто́ит дороже, но для работы многих современных программ требуются большие объемы памяти. Особенно это относится к вычислительным программам, предназначенным для решения сложных задач. Возникает проблема, для решения которой используется иерархическая организация памяти. В такой иерархии можно выделить два уровня — маленькую, но быструю память, и медленную память большего объема. К быстрой памяти относятся регистры процессора, кэш-память первого и второго уровня, а к медленной — главная и внешняя память (рис. 1.8).

На вершине иерархии находятся регистры процессора со временем доступа порядка наносекунд. Емкость одного регистра обычно несколько десятков битов (например, 32 бита). Далее следует кэш-память первого уровня объемом до нескольких десятков килобайтов и временем доступа около десяти наносекунд. Следующий уровень иерархии — кэш-память второго уровня величиной несколько сот килобайтов и временем доступа в несколько раз большим, чем у кэш-памяти первого уровня. Оперативная память имеет объем несколько десятков и сотен мегабайтов и время доступа около сотни наносекунд. Затем следует внешняя память объемом в десятки гигабайтов и временем доступа в десятки миллисекунд.

Рис. 1.8. Иерархия памяти компьютера

Иерархическая организация памяти должна обеспечить доступность необходимых данных тогда, когда они понадобятся, причем с наибольшей скоростью доступа, которую может обеспечить только верхний уровень иерархии. Данные, находящиеся в регистрах, контролируются транслятором или программистом, если он программирует на ассемблере. Содержимое других уровней иерархии контролируется автоматически — кэш-памяти аппаратно, оперативной и внешней памяти операционной системой с участием аппаратуры.

Кэш-память — это быстрая память небольшого объема, содержащая информацию из оперативной памяти, которая использовалась недавно. Идея кэш-памяти основана на *принципе локальности*. Принцип локальности состоит в том, что если программа обратилась к какому-то элементу данных, этот элемент может вскоре понадобиться вновь. Это локальность во времени. Кроме того, вскоре могут потребоваться элементы данных, адреса которых мало отличаются от адреса данных, используемых в настоящий момент. Это локальность в пространстве.

Система сама решает, какую информацию следует сохранить в кэш-памяти. Для этого используются разные алгоритмы. Схема взаимодействия кэш-памяти с главной памятью приведена на рис. 1.9.

Общая производительность вычислительной системы зависит от доли запросов к памяти, которая может быть обеспечена данными из кэш-памяти. В лучшем случае эта доля составляет несколько процентов. Распределение памяти и управление кэшированием возлагается на транслятор, программисту не надо об этом заботиться.