В. В. Воеводин, Вл. В. Воеводин

Параллельные вычисления

Рекомендовано Министерством образования Российской Федерации в качестве учебного пособия для студентов высших учебных заведений, обучающихся по направлению 510200 "Прикладная математика и информатика"

Санкт-Петербург
«БХВ-Петербург»
2002

УДК 681.3.06+519.68
ББК 32.973
В63

Воеводин В. В., Воеводин Вл. В.

В63 Параллельные вычисления. — СПб.: БХВ-Петербург, 2002. — 608 с.: ил.

ISBN 5-94157-160-7

Книга известных российских ученых посвящена обсуждению ключевых проблем современных параллельных вычислений. С единых позиций рассматриваются архитектуры параллельных вычислительных систем, технологии параллельного программирования, численные методы решения задач. Вместе со строгим описанием основных положений теории информационной структуры программ и алгоритмов, книга содержит богатый справочный материал, необходимый для организации эффективного решения больших задач на компьютерах с параллельной архитектурой.

Для научных работников, инженеров, преподавателей, аспирантов и студентов естественнонаучных специальностей

УДК 681.3.06+519.68
ББК 32.973

Группа подготовки издания:

Главный редактор	*Екатерина Кондукова*
Зам. главного редактора	*Анатолий Адаменко*
Зав. редакцией	*Анна Кузьмина*
Редактор	*Петр Науменко*
Компьютерная верстка	*Натальи Смирновой*
Корректор	*Наталия Першакова*
Дизайн обложки	*Игоря Цырульникова*
Зав. производством	*Николай Тверских*

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 23.09.02.
Формат 70×100 1/16. Печать офсетная. Усл. печ. л. 49,02.
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 5-94157-160-7

Содержание

Предисловие

Много ли надо знать о параллельных вычислениях?

Ничего нельзя сказать о глубине
лужи, если не попасть в нее.

Из законов Мерфи

В активе человечества имеется не так много изобретений, которые, едва возникнув, быстро распространяются по всему миру. Одним из них является компьютер. Появившись в середине двадцатого столетия, он через несколько десятков лет повсеместно стал незаменимым инструментом и надежным помощником в обработке самой разнообразной информации: цифровой, текстовой, визуальной, звуковой и др.

Известно, что первопричиной создания компьютеров была настоятельная необходимость быстрого проведения вычислительных работ, связанных с решением больших научно-технических задач в атомной физике, авиастроении, климатологии и т. п. Решение таких задач и сейчас остается главным стимулом совершенствования компьютеров. Однако в настоящее время основная сфера их применения связана с совсем иной деятельностью, в которой вычислительная составляющая либо отсутствует совсем, либо занимает небольшую часть.

Работа биржи, управление производством, офисные приложения, корпоративные информационные системы, процессы образования, игры, ведение домашнего хозяйства — это всего лишь отдельные примеры областей невычислительного использования компьютеров. В подобных областях вполне приемлемо применение персональных компьютеров и рабочих станций для достижения необходимых целей. Относительная простота процесса использования и комфортная программная среда формируют здесь устойчивое впечатление о компьютере как о весьма дружественном помощнике. И мало кто из пользователей такой техники догадывается и, тем более, знает, что все радикально меняется с переходом к решению больших и очень больших задач. Компьютеры становятся весьма сложными, куда-то пропадает дружественность интерфейса, программная среда переходит на жесткий командный язык и начинает требовать от пользователей предоставления такой информации, которая не всегда известна, и т. п.

Эта книга о больших и супербольших компьютерах и особенностях их использования. О тех проблемах, с которыми неизбежно приходится сталки-

ваться любому пользователю, вынужденному применять вычислительную технику на пределе ее возможностей.

Вообще говоря, большие задачи сами по себе не являются предметом нашего обсуждения, тем более, детального. Но иногда мы будем обращаться к ним явно, чтобы проиллюстрировать какие-то положения. Неявно же большие задачи буквально пронизывают всю книгу, т. к. любое обсуждение чаще всего предполагает, что решается именно большая задача. Формально характер обсуждения никак не связан с типом задач, хотя может ощущаться некоторая ориентация на научно-техническую сферу деятельности. Это объясняется всего лишь тем, что основное применение больших и супербольших компьютеров связано именно с данной сферой и потребности данной сферы оказывают наибольшее влияние на развитие компьютеров, по крайней мере, в настоящее время. Тем не менее, обсуждаемые в книге проблемы в одинаковой мере относятся к большим задачам как научно-техническим, так и любым другим, например, информационным.

Все, что связано с большими компьютерами и большими задачами, сопровождается характерным словом "параллельный": параллельные компьютеры, параллельные вычислительные системы, языки параллельного программирования, параллельные численные методы и т. п. В широкое употребление этот термин вошел почти сразу после появления первых компьютеров. Точнее, почти сразу после осознания того факта, что созданные компьютеры не в состоянии решить за приемлемое время многие задачи. Выход из создавшегося положения напрашивался сам собой. Если один компьютер не справляется с решением задачи за нужное время, то попробуем взять два, три, десять компьютеров и заставим их *одновременно* работать над различными частями общей задачи, надеясь получить соответствующее ускорение. Идея показалась плодотворной, и в научных исследованиях конкретное число объединяемых компьютеров довольно быстро превратилось в произвольное и даже сколь угодно большое число.

Объединение компьютеров в единую систему потянуло за собой множество следствий. Чтобы обеспечить отдельные компьютеры работой, необходимо исходную задачу разделить на фрагменты, которые можно выполнять *независимо* друг от друга. Так стали возникать специальные численные методы, допускающие возможность подобного разделения. Чтобы описать возможность одновременного выполнения разных фрагментов задачи на разных компьютерах, потребовались специальные языки программирования, специальные операционные системы и т. д. Постепенно такие слова, как "одновременный", "независимый" и похожие на них стали заменяться одним словом *"параллельный"*. Всё это синонимы, если иметь в виду описание каких-то процессов, действий, фактов, состояний, не связанных друг с другом. Ничего иного слова "параллелизм" и "параллельный" в областях, относящихся к компьютерам, не означают.

Далеко не сразу удалось объединить большое число компьютеров. Первые компьютеры были слишком громоздкими, потребляли слишком много энергии, да и многие технологические проблемы комплексирования еще не нашли эффективного решения. Но со временем успехи микроэлектроники привели к тому, что важнейшие элементы компьютеров по многим своим параметрам, включая размеры и объем потребляемой энергии, стали меньше в тысячи и более раз. Идея объединения большого числа компьютеров в единую систему стала главенствовать в повышении общей производительности вычислительной техники. В одной из самых больших современных систем ASCI White объединено 8192 процессора. При этом достигаются весьма впечатляющие суммарные характеристики: пиковая производительность более 12 Тфлопс, оперативная память 4 Тбайт, дисковый массив 160 Тбайт. Но всем этим богатством нужно еще уметь воспользоваться.

Параллелизм на различных уровнях характерен для всех современных компьютеров от персональных до супербольших: одновременно функционирует множество процессоров, передаются данные по коммуникационной сети, работают устройства ввода/вывода, осуществляются другие действия. Любой параллелизм направлен на повышение эффективности работы компьютера. Некоторые его виды реализованы жестко в "железе" или обслуживающих программах и недоступны для воздействия на него рядовому пользователю. Но с помощью жесткой реализации не удается достичь наибольшей эффективности в большинстве случаев. Поэтому многие виды параллелизма реализуются в компьютерах гибко, и пользователю предоставляется возможность распоряжаться ими по собственному усмотрению.

Под термином "параллельные вычисления" как раз и понимается вся совокупность вопросов, относящихся к созданию ресурсов параллелизма в процессах решения задач и гибкому управлению реализацией этого параллелизма с целью достижения наибольшей эффективности использования вычислительной техники.

Вот уже более полувека параллельные вычисления привлекают внимание самых разных специалистов. Три обстоятельства поддерживают к ним постоянный интерес. Во-первых, это очень важная сфера деятельности. Занимаясь параллельными вычислениями, исследователь понимает, что он делает что-то, относящееся к самым большим задачам, самым большим компьютерам и, следовательно, находящееся на передовом фронте науки. Как минимум, близость к передовому фронту науки вдохновляет. Во-вторых, это очень обширная сфера деятельности. Она затрагивает разработку численных методов, изучение структурных свойств алгоритмов, создание новых языков программирования и многое другое, связанное с интерфейсом между пользователем и собственно компьютером. Параллельные вычисления тесно связаны и с самим процессом конструирования вычислительной техники. Структура алгоритмов подсказывает необходимость внесения в компьютер изменений, эффективно поддерживающих реализацию структур-

ных особенностей. Инженерные же новшества стимулируют разработку новых алгоритмов, эффективно эти новшества использующих. И, наконец, в-третьих. С формальных позиций рассматриваемая сфера деятельности легко доступна для исследований. Достаточно более или менее познакомиться с ее основами на уровне популярной литературы и уже можно делать содержательные выводы, возможно, даже никем не опубликованные.

Последнее обстоятельство придает характерную окраску исследованиям в области параллельных вычислений. Легкое освоение основных положений привлекает к параллельным вычислениям большое число специалистов из других областей. Это порождает немало новых задач, особенно на стыке вычислительной техники и приложений. Некоторые из них оказываются интересными и перспективными. Но, с другой стороны, простота освоения основ порождает массу легковесных результатов, что не делает параллельные вычисления привлекательной областью для серьезных исследований и не позволяет раскрыть все их богатство и многообразие связей.

В подтверждение сказанного приведем некоторые данные на начало 80-х годов прошедшего столетия. Это был период, когда в широкое использование стали поступать самые разнообразные компьютеры и вычислительные системы параллельной архитектуры. К этому времени только по проблемам их освоения было опубликовано более 5000 работ, и поток публикаций имел явную тенденцию к расширению. В данном потоке заметную часть составляли работы, связанные с обсуждением особенностей реализации параллельных численных методов, главным образом, по линейной алгебре. Интерес к методам линейной алгебры вполне понятен, т. к. они составляют основу процессов решения многих сложных проблем. Согласно данным, взятым из библиографических указателей на начало 80-х годов XX века, в области вычислительных методов линейной алгебры было опубликовано примерно 8000 работ. При этом около 120 ученых мира имели по этой тематике более чем по 10 работ. Казалось бы, что именно им и развивать параллельные методы линейной алгебры. Но среди них на тот период лишь около 10 человек имели по одной-две публикации, косвенно относящихся к параллельным численным методам и их применению на параллельных вычислительных системах. И только несколько человек имели публикации, прямо посвященные обсуждению этих проблем.

Интересно отметить, что в определенном смысле история повторяется. Аналогичной была ситуация на рубеже появления первых вычислительных машин. Тогда математики-алгоритмисты также принимали слабое участие в разработке численных методов и программ для компьютеров. Прошло много лет, прежде чем положение изменилось. Последствия же этого периода ощущаются до сих пор. В настоящее время вычислительное сообщество активно переходит на использование кластеров, больших многопроцессорных систем, неоднородных систем и систем, распределенных по значительной территории. И опять ведущая роль в освоении новейшей техники в целом

принадлежит не профессионалам в создании численных методов, а разработчикам языков программирования, компиляторов, операционных систем и просто лицам, которым необходимо решать задачи. К тому времени, когда математиками будет понято, как надо правильно реализовывать численные методы на новой технике, менять программную среду, скорее всего, будет поздно. Но именно от численных методов во многом зависит, насколько успешно используется вычислительная техника.

Планируя написание этой книги, мы понимали, что заведомо придется искать компромисс между многообразием сторон затрагиваемой проблемы и полнотой их описания. Удачен выбранный компромисс или нет — судить читателю. Мы лишь хотим сначала пояснить свою позицию. Возможно, что после этого будет легче понять как цели появления того или иного материала в книге, так и пути их достижения.

У разных специалистов, связанных с параллельной вычислительной техникой, разные взгляды на то, что представляют собой параллельные вычисления, где и какие акценты надо ставить при изложении материала. В этой книге мы попытались отразить позицию пользователей и те проблемы, с которыми им приходится сталкиваться. Говоря о пользователях, мы имеем в виду, прежде всего, тех из них, которые осваивают *серийно* выпускаемую технику, которые вынуждены изучать особенности этой техники, чтобы решать на ней реальные задачи более *эффективно*, и которым для достижения эффективности приходится многократно *переписывать* свои программы.

Это в значительной мере определило представленный в книге материал. Мы сразу отказались от искушения описать многие, в том числе, весьма красивые инженерные, программистские и математические идеи, оказавшиеся по каким-либо причинам либо нереализованными, либо апробированными недостаточно. Но даже при таком ограничении интересных идей и достижений, относящихся к параллельным вычислениям, оказалось немало. Отбор материала и структура его описания подчинены одной из целей книги — показу многообразия и глубины связей различных направлений деятельности в параллельных вычислениях с соседними областями. В первую очередь, с созданием вычислительной техники, разработкой программного обеспечения и математическими исследованиями.

С какой бы стороны не рассматривать параллельную вычислительную технику, главным стимулом ее развития было и остается повышение эффективности процессов решения больших и очень больших задач. Эффективность зависит от производительности компьютеров, размеров и структуры их памяти, пропускной способности каналов связи. Но в не меньшей, если не большей, степени она зависит также от уровня развития языков программирования, компиляторов, операционных систем, численных методов и многих сопутствующих математических исследований. Если с этой точки

зрения взглянуть на приоритеты пользователей, то они всегда связаны с выбором тех средств, которые позволяют решать задачи более эффективно.

Эффективность — понятие многоплановое. Это удобство использования техники и программного обеспечения, наличие необходимого интерфейса, простота доступа и многое другое. Но главное — это достижение близкой к пиковой производительности компьютеров. Данный фактор настолько важен, что всю историю развития вычислительной техники и связанных с ней областей можно описать как историю погони за наивысшей эффективностью решения задач.

Конечно, такой взгляд отражает точку зрения пользователей. Но ведь именно пользователям приходится "выжимать" все возможное из имеющихся у них средств и приводить в действие все "рычаги", чтобы достичь максимальной производительности компьютеров на своих конкретных задачах. Поэтому им нужно знать, где находятся явные и скрытые возможности повышения производительности и как наилучшим образом ими воспользоваться. Реальная производительность сложным образом зависит от всех составляющих процесса решения задач. Можно иметь высокопроизводительный компьютер. Но если компилятор создает не эффективный код, реальная производительность будет малой. Она будет малой и в том случае, если не эффективны используемые алгоритмы. Не эффективно работающая программа — это прямые потери производительности компьютера, средств на его приобретение, усилий на освоение и т. п. Таких потерь хотелось бы избежать или, по крайней мере, их минимизировать.

Проблемы пользователей нам известны не понаслышке. За плечами лежит более двадцати лет научной, производственной и педагогической деятельности в области параллельных вычислений. За это время освоены многие компьютеры и большие вычислительные системы. Было решено немало задач. Но не было ни одного случая, когда одна и та же программа эффективно реализовывалась без существенной переделки при переходе к другой технике. Конечно, хотя и трудно, но все же можно заново переписать программу, удовлетворяя требованиям языка программирования и штатного компилятора. Однако новую большую программу суперсложно сделать эффективно работающей.

Технология обнаружения узких мест процесса реализации программы плохо алгоритмизирована. Опыт показывает, что для повышения эффективности приходится рассматривать все этапы действий, начиная от постановки задачи и кончая изучением архитектуры компьютера, через выбор численного метода и алгоритма, тщательно учитывая особенности языка программирования и даже компилятора. Основной способ обнаружения узких мест — это метод проб и ошибок, сопровождающийся большим числом прогонов вариантов программы. Необходимость многих экспериментов объясняется скудностью информации, получаемой от компилятора после каждого прогона. К тому же

ни один компилятор не дает никаких гарантий относительно меры эффективности реализуемых им отдельных конструкций языка программирования. С самого начала пользователь ставится в такие условия, когда он не знает, как надо писать эффективные программы. Получить соответствующую информацию он может только на основе опыта, изучая почти как черный ящик влияние различных форм описания алгоритмов на эффективность.

Наши исследования показывают, что большинство из узких мест может быть объединено в три группы. Первая связана собственно с компьютером. На любом параллельном компьютере не все потоки данных обрабатываются одинаково. Какие-то из них реализуются предельно эффективно, какие-то достаточно плохо. Изучая особенности архитектуры компьютера, очень важно понять, что представляет собой те и другие потоки и описать их математически. Вторая определяется структурой связей между операциями программы. Не в каждой программе обязаны существовать фрагменты, реализуемые эффективно на конкретном компьютере. И, наконец, третья зависит от используемой в компиляторе технологии отображения программ в машинный код.

Если технология позволяет разложить программу на фрагменты, по которым почти всегда будет строиться эффективный код, то узких мест в этой группе может не быть. Такие технологии были разработаны для первых параллельных компьютеров. Но по мере усложнения вычислительной техники технологии компиляции становятся все менее и менее эффективными, и узких мест в третьей группе оказывается все больше и больше. Для больших распределенных систем узкие места компиляции начинают играть решающую роль в потере общей эффективности. Поэтому уже давно наметилось и теперь почти воплотилось в реальность "компромиссное" разделение труда: наименее алгоритмизированные и сложно реализуемые этапы компиляции, в первую очередь, расщепление программы на параллельные ветви вычислений, распределение данных по модулям памяти и организацию пересылок данных возлагаются на пользователя. Проблем от этого не стало меньше. Просто о том, о чем раньше заботились разработчики компиляторов, теперь должны беспокоиться сами пользователи.

Акцентированное внимание к узким местам процесса решения задач является характерной чертой настоящей книги. Узкие места описываются и изучаются практически всюду: в параллельных компьютерах и больших вычислительных системах, процессах работы многих функциональных устройств, конструировании численных методов, языках и системах программирования, различных приложениях и даже в работе пользователей. Особое внимание уделяется узким местам, связанным с выявлением параллельных структур алгоритмов и программ и их отображением на архитектуру компьютеров.

Исследуя различные вопросы параллельных вычислений, мы неоднократно убеждались в том, что все они связаны многими незримыми нитями как

между собой, так и с совершенно далекими от любого параллелизма областями знаний. Эти связи не лежат на поверхности, и почувствовать их можно, лишь поставив исследования на строгую математическую основу. И тогда начинает возникать ощущение, что параллельные вычисления своими корнями уходят во что-то очень фундаментальное. Возможно, это что-то есть информационная структура алгоритмов. Возможно, оно окажется чем-то иным. Но в любом случае появляется уверенность, что глубокое изучение параллельных вычислений приведет, в конце концов, к пониманию того, как должны быть устроены наши алгоритмы, компьютеры и программное обеспечение. Сейчас параллельные вычисления представляются гигантским айсбергом, верхушка которого исследована весьма основательно. Исследование же остальной его части только начинается. Перспективу этого процесса мы также попытались отразить в нашей книге.

Содержание книги и ее структура хорошо видны из подробного содержания. Поэтому сделаем по ней лишь несколько замечаний. В своей основе книга является монографией, поскольку в нее включено много уникального авторского материала. Однако содержание систематизировано таким образом, что книга может служить учебным пособием для студентов, аспирантов и специалистов, связанных в своей деятельности с направлением "Прикладная математика и информатика". Для ее чтения нужны минимальные начальные знания. Некоторые разделы доступны даже школьникам, владеющим навыками работы с компьютером. Вместе с тем, в книге также приводятся сведения очень высокой степени сложности, достойные внимания самых серьезных исследователей. Для облегчения знакомства с книгой каждая глава начинается с небольшого введения, поясняющего цель и особенности изложенного в ней материала. После ознакомления с новым материалом читателю предлагается ответить на вопросы и выполнить задания. Вопросы и задания, помеченные одной или двумя звездочками, относятся соответственно к сложным и очень сложным.

Создание учебного пособия широкого профиля по параллельным вычислениям — это, пожалуй, самая главная цель написания настоящей книги. Мы постарались с единых позиций не только рассказать об основах данного научного направления, но и показать его проблемы и перспективы развития. Конечно, по параллельным вычислениям имеется немало хороших книг и публикаций. Но все они, на наш взгляд, слишком фрагментарны и не создают цельного впечатления о рассматриваемой сфере деятельности, осуществляемой на стыке многих наук. Структура и форма изложения материала этой книги формировались довольно долго. Научные исследования позволили выделить опорные точки параллельных вычислений среди огромного числа общих результатов и наметить их связь между собой. В предварительных публикациях была предпринята попытка развить и систематизировать эти связи на основе строгих математических определений и выкладок. Различные фрагменты книги неоднократно обсуждались на конференциях, се-

минарах и в дискуссиях с коллегами. На основе отобранного материала в течение многих лет читаются основные и специальные курсы в Московском государственном университете им. М. В. Ломоносова, физико-техническом институте и международном государственном университете в г. Дубна. Решающим обстоятельством, повлиявшим на ускорение процесса написания книги, стало наше участие в молодежных научных школах. Необходимость создания учебных пособий по параллельным вычислениям там ощущалась особенно остро.

Заниматься параллельными вычислениями довольно сложно. Эту область за короткое время трудно освоить и, тем более, трудно получить хорошие результаты. Поэтому в коллективах, где проводятся подобные работы, имеет большое значение создание атмосферы поддержки, заинтересованности и терпеливости. Нам приятно выразить признательность академику Г. И. Марчуку, который такую атмосферу создал в институте вычислительной математики Российской академии наук и в течение многих лет поддерживал это направление исследований.

Мы благодарны академику В. А. Садовничему за столь же благоприятную атмосферу в Московском университете и поддержку работ научно-исследовательского вычислительного центра МГУ по созданию крупнейшего центра по высокопроизводительным вычислениям в системе вузов России. Многолетнее сотрудничество ИВМ РАН и НИВЦ МГУ в области параллельных вычислений позволило приобрести бесценный опыт как в решении больших задач, так и в правильной расстановке акцентов в процессах образования и подготовки высококвалифицированных кадров. Появление этой книги есть одно из следствий данного сотрудничества.

В процессе написания книги нам пришлось контактировать со многими коллегами. Некоторые из них предоставили свои материалы, помогающие лучше осветить отдельные разделы. С другими было просто полезно поговорить, чтобы сформировать нужную точку зрения. Всем им мы выражаем нашу признательность. Особенно отмечаем участие академика В. П. Дымникова, член-корреспондента А. В. Забродина, докторов физико-математических наук С. М. Абрамова, А. Н. Андрианова, А. Л. Ластовецкого, В. А. Крюкова, А. Н. Томилина, кандидатов физико-математических наук К. Н. Ефимкина, Н. А. Коновалова.

Мы выражаем благодарность Российскому фонду фундаментальных исследований и Министерству промышленности, науки и технологий Российской Федерации. Многие результаты, о которых говорится в книге, были получены в рамках выполнения грантов от этих организаций. Без поддержки этих организаций написать такую книгу было бы гораздо сложнее.

Мы благодарим представительство компании Hewlett-Packard в России за предоставленные материалы, возможность использования ее вычислительной техники, за внимание и постоянный интерес к нашей работе.

Мы исключительно благодарны А. И. Караваеву, прекрасно подготовившему все иллюстрации, использованные в данной книге. Наша искренняя благодарность сотрудникам лаборатории параллельных информационных технологий НИВЦ МГУ: кандидату физико-математических наук А. С. Антонову, А. Н. Андрееву и С. А. Жуматию за множество ценных замечаний по рукописи.

И, наконец, нашу особую признательность мы хотим выразить Т. С. Гамаюновой и С. Н. Воеводиной. Без их самоотверженной работы по набору текста, его проверке, подготовке оригинал-макета и коррекции материала книга могла бы не появиться в срок.

Параллельные вычисления — перспективная и динамично изменяющаяся область научной и прикладной деятельности. Мы надеемся, что выбрать правильный путь в ней поможет эта книга.

Валентин В. Воеводин и Владимир В. Воеводин

ЧАСТЬ I

ПАРАЛЛЕЛЬНЫЕ ВЫЧИСЛИТЕЛЬНЫЕ СИСТЕМЫ

Глава 1

Что скрывают "обыкновенные" компьютеры

Внутри каждой большой задачи сидит
маленькая, пытающаяся пробиться наружу.

Из законов Мерфи

В своем развитии персональные компьютеры достигли высокого уровня совершенства. Они компактны, обладают большой скоростью выполнения заданий и, что особенно важно, достаточно просты в обращении. Все эти качества привели к их широкому использованию, в том числе среди лиц, не имеющих специальной компьютерной подготовки. По существу персональный компьютер становится таким же привычным устройством, как пылесос, кухонный комбайн и телевизор. Он остается удобным инструментом до тех пор, пока не приходится решать очень большие задачи. Конечно, в этом случае можно попытаться использовать самый быстрый персональный компьютер с максимально большой памятью. Можно также перейти к использованию рабочей станции. Однако увеличение возможностей на этом пути все же ограничено. И тогда приходится обращаться к суперкомпьютерам.

Согласно распространенному мнению, суперкомпьютер — это такой компьютер, у которого все самое-самое: супербольшая скорость, супербольшая память, супербольшая цена. Это действительно так. Но, переходя от персонального компьютера к суперкомпьютеру, пользователю приходится также сталкиваться с очень большими трудностями в их использовании. Главное, на суперкомпьютерах отсутствует характерная для персональных компьютеров пользовательская среда. Оказывается, что использовать суперкомпьютер гораздо сложнее, чем персональный компьютер, а некоторые суперкомпьютеры даже очень сложно. Кроме этого, выясняется, что разных типов суперкомпьютеров довольно много, они не совместимы друг с другом, и на разных суперкомпьютерах совсем разные пользовательские среды. Взвесив предстоящие трудности, пользователи нередко отказываются от использования суперкомпьютеров вообще или переходят к более простым суперкомпьютерам. Если же пользователь все же решается использовать суперкомпьютер, ему важно понимать причины появления новых трудностей и пути их преодоления.

Современные суперкомпьютеры возникли не вдруг. Они являются продуктом длительного развития "обыкновенных" компьютеров, к которым, кстати, относится и типовой персональный компьютер. Эти "обыкновенные" компью-

теры имеют ряд принципиальных узких мест, которые не позволяют достичь сколь угодно большой скорости и иметь сколь угодно большую память. Стремление сделать компьютеры более мощными и развязать узкие места привело, в конце концов, к тем изменениям, которые превратили "обыкновенный" компьютер в суперкомпьютер. Узких мест в компьютерах много, много и способов их развязки. Это породило много типов суперкомпьютеров. К сожалению, за возросшие скорости и объемы памяти приходится платить. Для пользователей эта плата связана, в первую очередь, со значительным усложнением и ухудшением пользовательской среды. Многое из того невидимого, что делали "обыкновенные" компьютеры, на суперкомпьютерах пользователям надо делать самим. Просто потому, что проблемы эффективного обслуживания пользовательских задач стали настолько сложными, что пока их не научились хорошо решать в автоматическом режиме.

Чтобы лучше понять причины появления новых трудностей и, возможно, наметить в дальнейшем пути их преодоления, рассмотрим сначала, как устроен "обыкновенный" компьютер. Мы не будем заниматься детальным и, тем более, инженерным анализом. Лишь рассмотрим те узкие места, развязывание которых, с точки зрения пользователя, и превращает "обыкновенный" компьютер в суперкомпьютер с его достоинствами и недостатками.

§ 1.1. Немного об устройстве компьютера

За очень редким исключением во всех существующих компьютерах, как "обыкновенных", так и супер, реализована одна и та же конструктивная идея. Она состоит в том, что вся исходная и перерабатываемая информация хранится в компьютерах в форме некоторого множества двоичных разрядов или *битов*, а любое ее преобразование сводится, в конце концов, к преобразованию битов с помощью нескольких простых функций.

Сейчас нереально даже представить такую ситуацию, при которой пользователь записывает и читает свою информацию в двоичном виде, а также описывает ее преобразование как преобразование множества битов. В действительности он всегда общается с компьютером на каком-нибудь приемлемом для него языке. Часто пользователь даже не знает, какие процессы происходят в конкретном компьютере при решении его задач. Однако для нас очень важно помнить, что, тем не менее, все эти процессы в любом случае осуществляются на битовом уровне. Частично переход к битовому уровню и обратно выполняется автоматически с помощью аппаратных средств, частично программным путем. От того, насколько эффективно реализуются эти переходы, в немалой степени зависит успех в решении пользовательских задач.

Не с каждого входного языка разработчики компьютеров и его программного окружения могут автоматически организовать эффективные бинарные процессы. И тогда они начинают требовать от пользователя какие-то допол-

нительные сведения. Чаще всего дополнительные требования формулируются в специфических компьютерных терминах, не всегда очевидным образом связанных с задачей или методом ее решения. Все это усложняет как входной язык, так и всю процедуру общения с компьютером. Такая ситуация особенно характерна для суперкомпьютеров, но не так уж редко она возникает и в связи с "обыкновенными" компьютерами.

Основным информационным элементом в компьютере является *слово*. Каждое слово представляет собой упорядоченный набор битов. Слово может делиться на *байты*. Байт означает упорядоченный набор из 8 битов. Число битов в слове называется его *длиной*. В любом конкретном компьютере все слова имеют одну и ту же длину. В разных компьютерах длина слов может быть разной. Например, в персональном компьютере слово состоит из одного байта, в компьютере Cray-1 оно состоит из 64 битов. Если в слове записана какая-то информация, то это означает, что в каждом бите слова зафиксировано одно из двух возможных его состояний 0 или 1. Совокупность состояний всех битов слова определяет *содержимое слова*. Устройство, в котором хранится все множество слов, обобщенно называется *памятью*. Оно может быть простым или сложным, однородным по устройству или разнородным в зависимости от типа или назначения компьютера. Все слова поименованы. Имя слова называется *адресом*. Структура адреса в определенном смысле отражает структуру памяти. Разные слова имеют разные адреса. Каждый адрес связан с конкретным физическим местом в памяти.

Если память устроена сложно и, к тому же, разнородна по своему строению, времена обращения к отдельным словам могут варьироваться в очень широких пределах, отличаясь друг от друга не только в разы, но и в десятки, сотни и даже тысячи раз. Это исключительно важное обстоятельство и оно нередко решающим образом определяет время решения задач. К различным вопросам эффективной работы с памятью мы будем возвращаться в этой книге неоднократно.

Основная задача любого компьютера состоит в преобразовании хранящейся в памяти информации. Она всегда реализуется как выполнение последовательности *простых* однозначных функций над содержимым отдельных слов. Как правило, все функции имеют не более двух аргументов. Функции могут использовать и/или изменять как слова целиком, так и их части. Вообще говоря, разные компьютеры могут иметь разные наборы исполняемых функций. Однако очень часто эти наборы функционально совпадают во многом или полностью, различаясь лишь техникой реализации. Наиболее распространенными функциями являются простейшие арифметические операции над числами (сложение, вычитание, умножение и т. д.) и логические операции булевой алгебры над битами слов (конъюнкция, дизъюнкция и т. д.). Обычно в компьютерной терминологии все функции называются *операциями*, а значение аргумента, иногда сам аргумент и даже адрес слова, где содержится аргумент, — *операндами*.

В компьютере операция реализуется в виде некоторой электронной схемы. Для обозначения реализаций этих схем используется самая различная терминология. Они называются чипом, устройством, процессором, сопроцессором и т. п. в зависимости от их сложности, связи друг с другом и даже вкуса конструктора. Мы будем использовать термин *"устройство"*, оставляя другие названия для иных целей. Совокупность устройств, реализующих арифметические и логические операции, называется *арифметико-логическим устройством* (АЛУ).

Кроме операций над содержанием памяти, компьютер должен осуществлять также и все действия, связанные с организацией процесса преобразования информации. Возможные действия компьютера описываются системой *машинных команд*. Как и любая другая информация, машинная команда записывается как содержимое слова. В описании команды задаются код операции и те операнды, над которыми операция будет проводить действия. Некоторые машинные команды выполняют действия над словами, расположенными в строго определенном месте, например, в фиксированных регистрах. Такие команды не требуют явного указания операндов. Система команд всегда устроена таким образом, что выполнение каждой команды однозначно определяет следующую команду. Любой процесс преобразования информации описывается конкретной совокупностью машинных команд из числа возможных для данного компьютера. Эта совокупность называется *машинным кодом* или программой во *внутреннем коде*. Тем самым подчеркивается, что процесс описан программой на языке машинных команд, а не как-либо иначе.

Структура машинных команд всегда привязана к структуре компьютера и может быть совсем не похожей на структуру тех действий, которые пользователь предполагает выполнять. Как правило, свои действия пользователь описывает программами на *языках высокого уровня*. Компьютеры их "не понимают". Поэтому, чтобы программы могли быть выполнены, они должны быть переведены, прежде всего, в *эквивалентный* машинный код. Такой перевод осуществляют специальные программные системы, называемые *компиляторами*. Компиляторы очень сложны и выполняют очень большой объем работы. От того, как именно они преобразуют программы пользователя, решающим образом зависит эффективность реализации получаемого машинного кода и, следовательно, эффективность процесса решения задач.

До того как машинный код начнет исполняться, все команды и необходимые данные должны быть помещены или, как говорят, загружены в память. Это осуществляется с помощью специальных команд через так называемые устройства *ввода*. К ним относятся устройства считывания информации с дискеты и лазерных дисков, сканеры, клавиатура и т. п. Результаты работы компьютера выводятся из памяти также с помощью специальных команд через устройства *вывода*. К ним относятся устройства записи информации на дискеты и лазерные диски, графопостроители, принтеры и т. п. Сейчас

существует огромное разнообразие устройств ввода/вывода (УВВ). Но независимо от их конструктивных особенностей УВВ остаются самыми медленными устройствами компьютера. Это обстоятельство вызывает много трудных проблем, когда объем входной или выходной информации оказывается очень большим.

После загрузки в память машинного кода и данных компьютер может начинать свою работу. Руководство ею осуществляет устройство *управления* (УУ). Оно содержит необходимые регистры, счетчики и другие элементы, обеспечивающие управление перемещением информации между памятью, АЛУ, УВВ и другими частями компьютера. УУ должно решать две противоречивые задачи. С одной стороны, УУ должно работать достаточно быстро, чтобы не задерживать процесс решения задачи. С другой стороны, УУ должно управлять самыми различными устройствами, в том числе, удаленными друг от друга и работающими одновременно. Поэтому УУ всегда устроено по иерархическому и распределенному принципу. Самый быстрый уровень — это те электронные схемы, которые расшифровывают содержание очередной команды, выделяют код операции и адреса операторов, выбирают для анализа одну или несколько последующих команд. Другой уровень УУ — это, например, схемы, управляющие выполнением операций АЛУ. Совокупность устройства управления, арифметико-логического устройства и блока наиболее быстрой памяти называется *центральным процессором*.

Во всех современных компьютерах реализуются два способа управления: схемный и микропрограммный. Схемный используется тогда, когда возникает необходимость очень быстрого, но относительно простого управления. Микропрограммный — в случае сложного управления. В целом УУ занимает центральное место в управлении работой компьютера. Но оно обросло многими программными и микропрограммными системами, связанными с решением частных управленческих задач. Некоторые из них очень сложны. Например, система управления файлами по существу представляет собой специализированную вычислительную машину, которая имеет свою память и связана с памятью и магнитными дисками основного компьютера. Еще более сложной является система управления графической информацией и т. п. Совокупность программных систем, обеспечивающих управление работой компьютера, принято называть *операционной системой*. Для всех современных компьютеров операционная система неотделима от собственно компьютера и является его программным продолжением. Один и тот же компьютер может иметь несколько операционных систем. Например, для персональных компьютеров широко распространены операционные системы MS-DOS, Windows и Unix. Каждая из них имеет свои особенности и решает свои задачи.

Такова общая схема "обыкновенного" компьютера. Она представлена на рис 1.1. Конечно, эта схема идеализирована и лишь иллюстрирует связи между отдельными частями компьютера. Тем не менее, в ней отражено все то, что

мы будем обсуждать более детально, выявляя "узкие места". Отметим, что компьютеры подобного типа кроме названия "обыкновенный" имеют также названия "однопроцессорный", "фон-неймановской архитектуры" и др.

Перейдем теперь к более детальному обсуждению отдельных элементов "обыкновенного" компьютера.

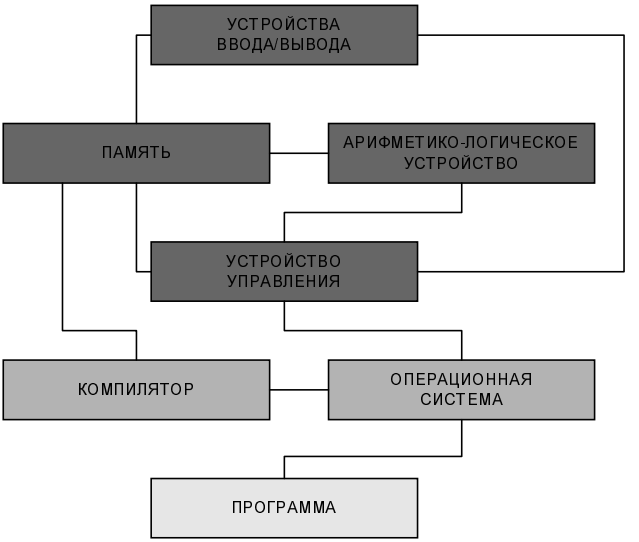

Рис. 1.1. Общая схема компьютера

Вопросы и задания

1. Приведите примеры физических, электрических и механических устройств, имеющих $p = 2, 3, 8, 10$ устойчивых состояний.
2. На основе предложенных устройств постройте схемы выполнения арифметических операций над целыми числами.
3. Исследуйте любой калькулятор как компьютер. Каковы в нем центральный процессор, память, операционная система, форма представления чисел?
4. Предположим, что компьютер имеет универсальный процессор и память объемом в 1 или 2 ячейки. Какие задачи можно решать на таком компьютере?
5. Знаете ли вы, что первые лекции об автоматизации процесса вычислений прочитаны английским математиком и экономистом Ч. Бэббиджем в 1840 г.?
6. Условный переход является одной из важнейших команд управления. Знаете ли вы, что впервые идея использовать условные переходы была высказана ирландским бухгалтером П. Лудгейтом в 1903 г.?

7. Рассмотрите любую систему машинных команд. С помощью команд, реализующих логические операции и операции безусловного перехода, составьте программу, реализующую операцию условного перехода.
8. Знаете ли вы, что первая вычислительная машина была построена немецким ученым К. Цузе в 1936 г. и была она механической?
9. Знаете ли вы, что первая отечественная электронная вычислительная машина МЭСМ была построена коллективом, возглавляемым академиком С. А. Лебедевым, в 1951 г.?

§ 1.2. Операции с числами

Есть одно обстоятельство, общее для всех компьютеров, осуществляющих работу с числами. Это — формы представления чисел. С одной стороны, они диктуются природой самих чисел. С другой — битовым представлением информации.

Известно [5], что любое вещественное число $x \neq 0$ можно единственным образом представить в виде бесконечного ряда по степеням числа 2, т. е.

$$x = \pm \sum_{i=-\infty}^{b} a_i 2^i. \qquad (1.1)$$

Здесь коэффициенты a_i принимают одно из значений 0 или 1, число b зависит от числа x и $a_b = 1$. При хранении чисел a_i в компьютере их можно отождествлять с двоичными разрядами слова. Поэтому и коэффициенты a_i часто называют *двоичными разрядами числа x*. При этом коэффициент a_i называется i-ым разрядом. Вместо ряда (1.1) число x также записывают просто перечислением коэффициентов

$$x = \pm\, a_b a_{b-1} \ldots a_0, a_{-1} a_{-2} \ldots, \qquad (1.2)$$

иногда с указанием знака, иногда без него. Не всегда ставится и *запятая*, отделяющая коэффициенты при неотрицательных степенях двойки в (1.1) от коэффициентов при отрицательных степенях.

Никакие технические решения не дают возможность хранить все двоичные разряды числа, т. к. в общем случае их бесконечно много. Поэтому числа представляются лишь конечным набором *старших* разрядов. Другими словами, от числа (1.2) остается только набор коэффициентов $a_b \ldots a_s$ для некоторого $s \leq b$. Следовательно, любое число, имеющее в своем разложении (1.1) большое количество значащих разрядов, должно заменяться каким-то образом числом, имеющим отведенное количество разрядов. Эта операция замены называется *округлением чисел*. Разность между округленным и округляемым числами называется *ошибкой округления*.

Теоретически операция округления числа может быть либо однозначно определенной функцией округляемого числа, либо случайной функцией. Как случайная функция она реализуется в компьютерах исключительно редко, т. к. в этом случае *невозможно* при повторении расчетов получать одни и те же результаты. Как однозначно определенная функция операция округления может быть реализована самыми разными способами, к чему есть веские причины. Операция округления исключительно важна для пользователя. Чем меньше ошибка округления, тем в общем случае точнее результат. Наилучшее округление очевидно, и реализуется оно обычным "школьным" правилом. В этом случае ошибка округления никогда не превосходит половины последнего разряда в округленном числе. Но реализация данного правила в компьютере требует дополнительно и аппаратуры, и времени. В погоне за скоростью выполнения арифметических операций инженеры часто реализуют более простые операции округления. Иногда они делают это очень неудачно. Например, просто плохо была выполнена операция округления в первых моделях знаменитого компьютера Cray-1. Только резкая критика со стороны пользователей заставила изменить операцию округления в последующих моделях.

Конечно, ошибки округления по отношению к самим числам являются малыми поправками. При выполнении различных операций они вносятся в результат почти всегда. И эти малые поправки радикально меняют свойства операций. Если при точном выполнении операции сложения, вычитания, умножения и деления обладают свойствами ассоциативности, коммутативности и дистрибутивности, то эти свойства при выполнении тех же операций с округлением пропадают. Факт, который доставляет очень много хлопот при конструировании численных методов.

Представление чисел в виде двоичных разложений (1.1), безусловно, продиктовано двоичным представлением любой информации в компьютерах. К сожалению, оно имеет ряд неустранимых изъянов. Доказано, в частности, *что при выполнении таких арифметических операций, как сложение и вычитание, операция округления, реализованная по любому неслучайному правилу, дает ошибки с ненулевым смещением*. Это приводит к ненулевым смещениям ошибок и в окончательных результатах, о чем не следует забывать. С точки зрения ошибок округления более привлекательным является разложение чисел по степеням числа 3. Подробнее с округлением чисел можно познакомиться в книге [5].

Если интересно знать, как реализована операция округления на конкретном компьютере, проведите на нем следующий эксперимент. При точных вычислениях рекуррентная формула

$$y_0 = 1,\ y_k = (y_{k-1}/k)k,\ k \geq 1$$

дает $y_k = 1$ для всех k. Не изменяя положения скобок, проведите расчеты по ней на компьютере в течение 10—15 минут и выведите для некоторых k значения ошибки

$$\varepsilon_k = |\tilde{y}_k - 1|.$$

Здесь $\tilde{y}_k$ есть реально вычисленная величина. Скорее всего, ε_k будет линейно расти с ростом k. Но ведь некоторые задачи считаются часами, сутками и даже дольше. Теперь задайтесь вопросом о том, можно ли доверять полученному решению. Если возникли сомнения, вы на правильном пути. Тогда полистайте книгу [5] более внимательно.

Требование унифицированного выполнения арифметических операций приводит к унификации изображения чисел в компьютерах. Пусть для каждого числа отводится τ двоичных разрядов памяти. Обычно они составляют слово или несколько слов. Прежде всего должно быть установлено взаимно однозначное соответствие между отдельными битами и двоичными разрядами числа. В зависимости от того, является ли это соответствие одним и тем же для всех чисел или зависит от числа, различают два основных способа представления чисел в компьютере, называемых соответственно представлением с фиксированной и плавающей запятой.

Предположим, что τ двоичных разрядов памяти служат для изображения τ последовательных разрядов чисел, причем положение запятой среди них фиксировано и является одним и тем же для всех чисел. Будем считать, что на изображение разрядов, стоящих слева от запятой, отводится r битов, где $r \geq 0$. Такой способ представления чисел называется представлением с *фиксированной запятой*. С помощью этого способа можно точно запомнить двоичное разложение любого из чисел, имеющих не более r ненулевых разрядов слева от запятой и не более $\tau - r$ ненулевых разрядов справа от запятой. Все эти числа лежат в диапазоне

$$-2^r < x < 2^r.$$

Один из недостатков представления чисел с фиксированной запятой виден сразу. Если число много меньше 2^r по модулю, то большая часть из отведенных τ битов изображает старшие нулевые разряды и фактически не используется. Поэтому аппроксимация малых чисел числом с фиксированной запятой связана с большой относительной ошибкой. Однако для чисел, близких к 2^r по модулю, используются все τ битов. В этом случае относительная ошибка является минимальной. Абсолютная ошибка представления чисел с фиксированной запятой всегда лежит в одних и тех же пределах независимо от величины чисел. Достоинством представления чисел с фиксированной запятой является относительная простота алгоритмов сложения и умножения. По существу они ничем не отличаются от хорошо известных "школьных" алгоритмов выполнения этих операций.

Представление чисел с *плавающей запятой* заключается в следующем. Принимая во внимание (1.1), запишем число x в виде

$$x = a2^b. \tag{1.3}$$

Тогда будем иметь, что

$$1/2 \le |a| < 1. \qquad (1.4)$$

Число a называется *мантиссой* числа x, число b — его двоичным *порядком*. Если $x = 0$, то считается, что $a = 0$, а число b не определено. Пусть на изображение порядка без знака отводится r битов, на изображение мантиссы без знака $\tau - r$ битов. Представив порядок и мантиссу как числа с фиксированной запятой, мы получаем представление числа x с плавающей запятой. При изображении мантиссы и порядка нет отмеченной выше потери относительной точности, т. к. в обоих случаях положение запятой строго определено. Порядок всегда является целым числом, а первый после запятой двоичный разряд мантиссы всегда равен 1. Порядок представляется точно. Мантисса будет точно представлена только для чисел, которые в двоичном разложении имеют не более $\tau - r$ ненулевых разрядов. Все числа, которые могут быть представлены с плавающей запятой, всегда представляются с высокой относительной точностью. Это числа из диапазона

$$2^{-2^r} \le |x| < 2^{2^r - 1}.$$

Минимальное положительное число ω, которое можно представить с плавающей запятой, называется *машинным нулем*. Ясно, что в нашей трактовке представления чисел

$$\omega = 2^{-2^r}.$$

Все числа из диапазона

$$-2^{-2^r} < x < 2^{-2^r}$$

изображаются нулем. В разных компьютерах под порядок и мантиссу отводятся разные числа разрядов. В частности, в упоминавшемся уже компьютере Cray-1 на мантиссу со знаком выделено 49 разрядов, на двоичный порядок — 15 разрядов. Огромный диапазон представляемых чисел и большая их относительная точность, безусловно, определены ориентацией компьютера Cray-1 на научно-технические расчеты.

В современных компьютерах используются оба способа представления чисел. Операции над числами с фиксированной запятой выполняются быстрее, чем над числами с плавающей запятой. Это связано с тем, что при реализации операций с плавающей запятой по существу приходится выполнять все действия с парами чисел с фиксированной запятой. Рассмотрим два числа

$$x = a2^b, \quad y = c2^d$$

с плавающей запятой. Имеем

$$xy = (ac)2^{(b+d)}.$$

Если $1/2 \le |ac| < 1$, то мы сразу получаем представление числа xy в форме с плавающей запятой. Но если $1/4 \le |ac| < 1/2$, то нужное представление числа xy будет другим. Именно,

$$xy = (2ac)2^{b+d-1}.$$

Естественно, что при реализации операции умножения чисел с плавающей запятой обе ситуации должны осуществляться несколько по-разному. Поэтому в процессе реализации операции умножения чисел должны быть ветвления.

Еще более сложно осуществляется операция сложения чисел с плавающей запятой. Пусть для определенности $b \ge d$. Имеем

$$(x + y) = (a + c2^{(d-b)})2^b, \tag{1.5}$$

причем всегда выполняется неравенство

$$|a + c2^{(d-b)}| < 2.$$

Если

$$|a + c2^{(d-b)}| = 0,$$

то представление (1.5) сразу дает форму числа $x + y$ с плавающей запятой. Предположим теперь, что

$$2^{r-1} \le |a + c2^{(d-b)}| < 2^r$$

для какого-нибудь целого числа r. Представим число $x + y$ в следующем виде:

$$x + y = ((a + c2^{(d-b)})2^{-r})2^{(b+r)}. \tag{1.6}$$

Так как выполняются соотношения

$$1/2 \le |(a + c2^{(d-b)})2^{-r}| < 1,$$

то (1.6) дает форму с плавающей запятой для числа $x + y$. Здесь $b + r$ есть порядок числа $x + y$, выражение

$$(a + c2^{(d-b)})2^{-r}$$

задает его мантиссу. Процесс реализации операции сложения также имеет ветвления, причем более сложные, чем у операции умножения.

Представление чисел с плавающей запятой оказывается предпочтительным в тех случаях, когда приходится иметь дело с числами из очень большого диапазона. Такая ситуация типична при проведении научно-технических расчетов. Однако довольно часто встречаются и другие ситуации. Например, в финансовых расчетах, задачах количественного учета заранее известен диапазон рассматриваемых чисел. Здесь вполне может использоваться представление чисел с фиксированной запятой. Необходимость экономии аппаратуры также приводит к фиксированной запятой. Умелое ее использование позволяет добиться большей скорости и большей точности решения задачи, чем использование плавающей запятой. Еще большего эффекта можно дос-

тичь путем разумного сочетания вычислений обоих типов. Однако мы не будем сколько-нибудь подробно останавливаться здесь на этих вопросах.

Устройства, выполняющие операции сложения/вычитания, называются *сумматорами*, выполняющие операции умножения — *умножителями*. Все эти устройства наряду с устройствами, реализующими логические операции, входят в состав АЛУ. Операции с фиксированной запятой выполняются быстрее аналогичных операций с плавающей запятой. Операции сложения выполняются быстрее операций умножения. Любые логические операции реализуются быстрее операций сложения/вычитания с фиксированной запятой. На каждом конкретном компьютере длительности выполнения стандартных операций отличаются друг от друга не более, чем в несколько раз.

Вопросы и задания

1. Докажите, что любое вещественное число x можно представить в виде троичного разложения

$$x = \sum_{i=-\infty}^{b} a_i 3^i,$$

где числа a_i принимают одно из значений 0, 1, −1.

2. Докажите, что знак числа x в п. 1 совпадает со знаком числа ab, если $ab \neq 0$.
3. Обозначим через x_s число, получаемое из троичного разложения числа x отбрасыванием всех разрядов, начиная с $(s-1)$-го. Докажите, что для троичного разложения $x_s > x$, если первый из ненулевых отброшенных разрядов отрицательный, и $x_s < x$, если он положительный.
4. Обладает ли двоичное разложение (1.1) аналогичным свойством?
5. Докажите, что число $(1/2)3^s$ не может быть задано конечной дробью по степеням числа 3.
6. Докажите, что в обозначениях пп. 3, 4 имеют место неулучшаемые оценки $|x - x_s| \le 2^s$ для двоичного разложения и $|x - x_s| \le (1/2)3^s$ для троичного разложения.
7. В какой системе, двоичной или троичной, легче выполнять операцию округления, если стремиться к минимизации абсолютной ошибки?
8. Знаете ли вы, что первая в мире электронная вычислительная машина "Сетунь", работающая на троичной системе, была сконструирована в Московском университете к.т.н. Н. П. Брусенцовым?
9. Предложите какой-либо способ изображения чисел, отличный от способа с фиксированной или плавающей запятой. Насколько сложно при этом выполняются операции над числами?
10. Почему числа из интервала $(-\omega, \omega)$, где ω есть машинный ноль, заменяются нулем, а не каким-либо другим числом?