1967
Компьютеры прежде и теперь

М. В. Уилкс

Кембриджский университет
Кембридж, Англия

Воспоминания о ранних разработках, приведших к появлению крупных электронных компьютеров, показывают, что для создания и внедрения в практику первых мощных и инженерно проработанных компьютеров потребовалось значительно больше времени, чем ожидалось. Замечания о современном состоянии компьютерной проблематики обосновывают необходимость дальнейших разработок.

Я не думаю, что многие из последующих тьюринговских лауреатов смогут похвастаться личным знакомством с Аланом Тьюрингом. Прославившая его работа о вычислимых числах была опубликована в 1936 г., когда компьютеров еще не существовало. Впоследствии Тьюринг стал одним из первых среди целой плеяды выдающихся математиков, внесших свой вклад в разработку и применение компьютеров. Он был весьма яркой фигурой на заре разработки цифрового компьютера в Англии, и было бы затруднительно не упоминать о нем, рассказывая об этом периоде.

ВРЕМЕНА ПЕРВООТКРЫВАТЕЛЕЙ

Самое важное событие в моей жизни произошло в 1946 г., когда я получил телеграмму с приглашением прослушать в конце лета того же года учебный курс по компьютерам в филадельфийской школе Мура по электротехнике. Мне удалось прослушать этот курс, хотя и не с самого начала, и он произвел на меня сильнейшее впечатление. Ничего подобного никогда раньше не было, а о достижениях школы Мура и других зачинателей компьютерной техники тогда знали лишь немногие. Курс слушали 28 человек из 20 организаций. В роли основных преподавателей выступали Джон Мочли и Проспер Экерт. Они находились на гребне успеха, создав первую электронную вычислительную машину ЭНИАК, работа которой, впрочем, основывалась не на принципе хранения программы в машинной памяти. Размер этой машины впечатляет даже теперь: она содержала примерно 18 000 электронных ламп. Хотя машина ЭНИАК весьма успешно и очень быстро справлялась с вычислением баллистических таблиц, для чего она и предназначалась,

но ее применение в качестве универсального вычислительного устройства наталкивалось на ряд серьезных ограничений. Во-первых, программа вводилась с помощью штекеров, гнезд и переключателей, и поэтому требовалось много времени для перехода от одной задачи к другой. Во-вторых, емкость внутренней памяти была рассчитана на хранение только 20 чисел. Экерт и Мочли сознавали, что основная проблема связана с памятью, и предлагали применять для будущих машин ультразвуковые линии задержки. При этом команды и числа находились бы смешанно в одной и той же памяти, привычным теперь для нас образом. Когда эти новые принципы были провозглашены, стало видно, что более мощные, чем ЭНИАК, компьютеры можно построить с использованием десятой части оборудования, затраченного на ЭНИАК.

В то время фон Нейман сотрудничал со школой Мура в качестве консультанта, но лично познакомиться с ним мне удалось лишь несколько позже. Вычислительная техника очень многим обязана фон Нейману. Он сразу осознал перспективы метода, получившего известность под названием «логическое проектирование», а также возможности, заложенные в принципе запоминаемой программы. Весьма существенно, что для поддержки этих новых идей фон Нейману пришлось воспользоваться своим громадным авторитетом и влиянием, поскольку некоторым они казались слишком революционными и раздавались громкие возгласы о том, что ультразвуковая память окажется недостаточно надежной и что смешивание команд и чисел в одной памяти противоестественно.

Последующие события убедительно подтвердили принципы, которым Экерт и Мочли научили в 1946 г. тех из нас, кому посчастливилось прослушать этот курс. Впрочем, в начале пятидесятых годов возникали затруднения. Разумеется, первые работоспособные компьютеры с запоминаемыми программами представляли собой лабораторные модели. Они не были в достаточной степени инженерно проработаны, и в них не в полной мере использовались технологические возможности того времени. Потребовалось неожиданно много времени для создания и внедрения в практику первых более мощных и полностью инженерно проработанных компьютеров. В ретроспективе этот период кажется не слишком длительным, но тогда это было время отчаянных поисков и даже взаимных обвинений.

В прошлом году я часто ощущал, что мы проходим весьма сходный этап в отношении разделения времени. Эта разработка влечет за собой много далеко идущих последствий относительно связей между компьютерами, с одной стороны, и отдельными пользователями и их сообществами — с другой, и она разожгла воображение многих людей. Прошло уже не-

сколько лет после того, как были продемонстрированы первые системы. И снова уходит неожиданно много времени на переход от экспериментальных систем к высокоразвитым, полностью воплотившим в себе современный уровень технологических возможностей. В результате настал период неуверенности и недоуменных вопросов, весьма напоминающий прежний период, о котором я упоминал. Когда все эти проблемы окажутся в прошлом, мы быстро забудем испытания и огорчения, которые теперь переживаем.

В ультразвуковых запоминающих устройствах было принято запоминать до 32 слов впритык друг к другу в одной и той же линии задержки. Частота импульсов была весьма высока, но все же пользователей очень раздражала пустая трата времени на ожидание, пока придет нужное слово. Поэтому большинство компьютеров на линиях задержки проектировались так, что программист, наловчившись, мог размещать свои команды и числа в памяти таким образом, чтобы минимизировать время ожидания. Сам Тьюринг был первооткрывателем этого способа логического проектирования. Позднее сходные методы применялись для компьютеров с памятью на магнитном барабане, и на основе этих методов расцвела общая теория так называемого *оптимального кодирования.* Я чувствовал, что этот род человеческой изобретательности бесперспективен в долгосрочном плане, поскольку рано или поздно мы должны были получить запоминающее устройство с поистине произвольным доступом. Поэтому нам в Кембридже совсем не пришлось заниматься оптимальным кодированием.

Профессия математика не помешала Тьюрингу проявлять определенный интерес к инженерной стороне проектирования компьютеров. В 1947 г. обсуждался вопрос о том, нельзя ли найти для ультразвуковых линий задержки материал, более дешевый, чем ртуть. Тьюринг внес в это обсуждение свежую струю, предложив использовать джин, который, по его словам, содержит алкоголь и воду именно в такой пропорции, чтобы обеспечить нулевой температурный коэффициент скорости прохождения при комнатной температуре.

Причина первоначальных успехов состояла в том, что группы в разных частях света готовились конструировать экспериментальные компьютеры, совсем необязательно стремясь сделать их прототипами для серийного производства. В результате формировались основы знаний о том, что работоспособно, а что неработоспособно, что выгодно делать, а что невыгодно. Хотя рассмотрение коммерчески доступных в настоящее время компьютеров не дает оснований считать, что эти уроки усвоены, несомненно, что такая широта фронта первоначальных исследований окупалась с лихвой. Я считаю важным, что-

бы такая широта фронта исследований имелась и теперь, когда мы учимся конструировать мультипрограммные микропроцессорные вычислительные системы с коллективным доступом. Вместо того чтобы собирать компоненты и электронные лампы для изготовления компьютера, мы должны теперь учиться собирать модули памяти, процессоры и периферийные устройства для изготовления системы. Я надеюсь, что найдутся деньги для конструирования больших систем только с исследовательскими целями.

Многие ранние разработки цифровых компьютеров выполнялись в университетах. Несколько лет назад было широко распространено мнение, что университеты уже сыграли свою роль в проектировании компьютеров и что теперь эту заботу можно благополучно предоставить промышленности. Я не считаю необходимым, чтобы проектированием компьютеров занимались все университеты, но меня радует, что некоторые из них сохранили активность в этой области деятельности. Помимо своих традиционных функций распространения знаний и сохранения открытой для общества информации, которая в противном случае могла бы быть скрыта, университеты в состоянии внести особый вклад, поскольку они не связаны коммерческими соображениями и, в частности, свободны от необходимости следовать моде.

ХОРОШИЙ И ПЛОХОЙ ЯЗЫКИ

Постепенно утихли споры относительно проектирования самих компьютеров, и мы стали обсуждать достоинства и недостатки изощренных приемов программирования. Началась битва за автоматизацию программирования, или, как мы сказали бы теперь, за использование языков программирования высокого уровня. Я хорошо помню свое участие в спорах на эту тему на одном из первых совещаний ACM примерно в 1953 г. Выступал и Джон Карр, который разделил программистов на две группы. Первая включала в себя «примитивов», которые полагали, что все команды следует писать в восьмеричном, шестнадцатеричном или неком сходном виде и не хотели тратить время на то, что они называли чудными схемами. Вторая состояла из «прогрессистов», считавших себя пионерами новой эры автоматизации программирования. Я поспешил отнести себя к прогрессистам, хотя, помнится, предостерегал против надежд на интерпретирующие системы, бывшие тогда в моде, и высказывался в пользу компиляторов. (Сомневаюсь, чтобы термин «компилятор» тогда широко использовался, хотя он был уже введен Грейс Хаппер.)

Серьезные возражения против автоматизации программирования должны были касаться эффективности. Компилированные программы не только работали медленнее, чем закодированные вручную, но, что имело тогда большее значение, они требовали больше памяти. Другими словами, для выполнения такой же работы нужен был более мощный компьютер. Все мы знаем, что, несмотря на справедливость этих возражений, они не оказались решающими и что люди пошли на эту жертву ради достижения автоматизации программирования. В сущности, без этого оказалось бы невозможным впечатляющее распространение вычислительной техники за последние несколько лет. Теперь ведется весьма сходная дискуссия относительно разделения времени, и возникающие возражения против такого разделения очень похожи на те возражения, которые ранее выдвигались против автоматизации программирования. И снова я нахожусь на стороне прогрессистов и надеюсь, что нынешние дебаты приведут к аналогичному результату.

Иногда я опасаюсь, что в дискуссии относительно автоматизации программирования симпатии Тьюринга определенно оказались бы на стороне «примитивов». Изобретенная им система программирования для первого компьютера в Манчестерском университете была в высшей степени изощренной. Он сам обладал исключительно гибким мышлением и не видел надобности делать уступки менее сообразительным людям. Я помню, как он решил, — вероятно, потому, что кто-то показал ему последовательность импульсов на осциллоскопе, — что нужно писать двоичные числа в обратном порядке, с последней значащей цифрой слева. При случае он распространял этот подход и на десятичную нотацию. Я хорошо помню, как однажды во время лекции, когда он перемножал на доске некоторые десятичные числа, чтобы проиллюстрировать тезис о проверке программ, все мы оказались не в состоянии следить за его мыслью, пока не поняли, что он пишет числа задом наперед. Не думаю, что он шутил или пытался посрамить нас; просто он не мог представить себе, что такой пустяк может так или иначе сказаться на чьем-то понимании сути дела.

Я полагаю, что через двадцать лет люди будут вспоминать период, в котором мы теперь живем, как время, когда только начиналось понимание основных принципов проектирования языков программирования. Меня огорчает, когда я слышу от вполне разумных людей мнение, что настало время выбрать в качестве всеобщего стандарта один или два языка. Нам действительно нужны временные стандарты для ориентации, но нам не следует рассчитывать на достижение стабильности еще по крайней мере некоторое время.

СИНТАКСИС БОЛЕЕ ВЫСОКОГО УРОВНЯ

Заметным достижением последних нескольких лет явилось существенное углубление понимания синтаксиса и синтаксического анализа. Оно привело к практическим успехам в конструировании компиляторов. Ранним достижением в этой области, не получившим в свое время должной оценки, был компилятор Брукера и Морриса.

Теперь люди начали понимать, что не все проблемы являются по своей природе лингвистическими и что настало время уделять больше внимания способам хранения данных в компьютерах, т. е. структурам данных. В своей прошлогодней Тьюринговской лекции А. Перлис привлек внимание к этой тематике. В настоящее время выбор языка программирования эквивалентен выбору структуры данных, и если эта структура не подходит для тех данных, которые вы хотите обрабатывать, то остается только глубоко посочувствовать вам. В некотором смысле представляется более логичным сначала выбрать подходящую для проблемы структуру данных, а затем искать или конструировать с помощью набора имеющихся средств язык, пригодный для манипулирования такой структурой данных. Люди иногда толкуют о языках программирования высокого и низкого уровней, не вполне отчетливо формулируя, что они при этом имеют в виду. Если язык высокого уровня таков, что структура данных фиксирована и не поддается изменениям, а в языке низкого уровня имеется некое разнообразие выбора структур данных, то я полагаю, что мы можем наблюдать отход к языкам программирования низкого уровня, по крайней мере для некоторых целей.

Впрочем, здесь я сделал бы замечание. В языке высокого уровня значительная часть синтаксиса и большая часть компилятора относятся к механизму формулирования описаний, к формированию составных операторов из простых и к казуистике условных операторов. Все это совершенно не зависит от того, что делают операторы, которые фактически работают с данными, или на что похожи структуры данных. В сущности, мы имеем два языка, один внутри другого. Внешний язык относится к управлению последовательностью действий, а внутренний язык оперирует данными. Можно было бы иметь стандартный внешний язык — или небольшое количество таких языков на выбор — и ряд внутренних языков, которые могли бы вкладываться в них. В случае необходимости, чтобы учесть особые обстоятельства, можно было бы сконструировать новый внутренний язык. При его встраивании во внешний он пользовался бы преимуществами мощных средств, обеспечиваемых внешним языком, для организации последовательности вычис-

лений. Когда я думаю о различных специальных языках, потребность в которых мы начали ощущать, — например, для управления в реальном времени, для машинной графики, написания операционных систем и др., — мне все больше кажется, что нам следует принять некую систему, которая избавила бы нас от необходимости затрачивать силы на разработку и освоение нового внешнего языка для каждой новой ситуации.

Фундаментальная важность структур данных может быть проиллюстрирована на примере рассмотрения проблемы проектирования единого языка, который оказывался бы предпочтительным как для чисто арифметической работы, так и для символьных манипуляций. Попытки создать такой язык оказались разочаровывающими. Трудность в том, что в этих двух случаях для эффективной реализации требуются совсем разные структуры данных. Возможно, нам следует признать эту трудность непреодолимой и отказаться от поисков такого языка общего пользования, который обеспечивал бы все для всех.

Существует тенденция развития программного обеспечения, которой, по-видимому, уделяется незаслуженно мало внимания. Речь идет о повышении мобильности, т. е. легкости переноса языковых систем с одного компьютера на другой. Долгое время имелась возможность обеспечивать такую мобильность посредством написания системы целиком на некотором широко используемом языке программирования высокого уровня, например на Алголе или Фортране. Однако этот метод заставляет применять структуры данных, свойственные языку, на котором написана система, что налагает очевидные ограничения на эффективность.

Для того чтобы систему можно было сразу переносить с одного компьютера на другой, не прибегая к фиксированному языку-посреднику, она должна быть написана в первую очередь в машинно-независимой форме. Здесь не место углубляться в различные приемы для переноса подходящим образом сконструированной системы. Они, в частности, включают начальную загрузку и использование примитивов и макроопределений. Часто обеспечение переноса включает в себя выполнение некоторой работы на компьютере, на котором система уже работает. Х. Хаски на самом раннем этапе успешно работал в этой области с системой Neliac.

Есть основания надеяться, что вновь найденная мобильность распространится на операционные системы или по крайней мере на их значительные части. В общем, мне кажется, что наступает новый период, когда в меньшей степени будет ощущаться неудобство несовместимости основных машинных кодов. Эта тенденция будет нарастать за счет расширения применения внутренних файловых систем, в которых информация

может храниться в системе в алфавитно-цифровой, т. е. полностью машинно-независимой форме. Тогда сможем без всяких затруднений присоединять к нашим компьютерным системам группы устройств, которые теперь считаются в принципе несовместимыми. В частности, я полагаю, что в больших системах будущего не обязательно все процессоры должны будут поставляться одним и тем же производителем.

ПРОЕКТИРОВАНИЕ И СБОРКА

Последние несколько лет характеризовались усиленным вниманием к машинной графике. Думаю, что специалисты по информатике уже давно понимали полезность при определенных обстоятельствах графических средств общения с компьютером, но для многих из нас явился неожиданным интерес к этой проблеме со стороны инженеров-механиков. Обычно инженеры общаются между собой с помощью чертежей и эскизов, и как только они увидели чертежи на экране монитора, многие из них сразу решили, что уже решены все проблемы использования компьютеров в инженерном проектировании. Разумеется, мы знаем, что дело обстоит совсем не так и что придется затратить много тяжкого труда, прежде чем удастся реализовать все возможности графического представления информации. Однако первая реакция инженеров показала нам два обстоятельства, о которых не следует забывать. Во-первых, по мнению инженеров-проектировщиков, обычные средства общения с компьютером оказываются совершенно неподходящими. Во-вторых, та или иная форма графического представления информации является чрезвычайно важной для современного машиностроительного конструирования. Мы должны либо обеспечить такую возможность в своих вычислительных системах, либо взять на себя непосильную задачу выучить новую породу инженеров, которым будет присущ другой способ мышления.

Имеются признаки того, что нынешний бум машинной графики будет сопровождаться соответствующим возрастанием интереса к компьютерному управлению объектами. Уже началась разработка нескольких таких проектов. Они в значительной степени стимулируются модой на искусственный интеллект. Эта мода отражается и в выбираемых задачах, и в применяемой стратегии программирования.

Впрочем, мои личные интересы в этой области являются более прагматичными. Я считаю, что компьютерно управляемые механические устройства имеют большое будущее на заводах и повсеместно. Автоматизация изготовления деталей машин уже достигла впечатляющих масштабов, и появление станков с числовым программным управлением сделало возможным

автоматическое изготовление весьма сложных деталей относительно малыми партиями. Напротив, гораздо более скромными являются успехи в автоматизации сборки деталей и узлов в готовые изделия.

Методы искусственного интеллекта могут оказаться менее подходящими для решения задач проектирования автоматизированных сборочных устройств. Животные и машины строятся из совсем разных материалов и на совершенно различных принципах. Когда инженеры пытаются почерпнуть вдохновение из изучения способов деятельности животных, они обычно идут по ложному пути. Это со всей очевидностью иллюстрируется историей ранних попыток конструировать летающие машины с машущими крыльями. По моему мнению, в самом недалеком будущем мы увидим компьютерно управляемые конвейерные линии, вдоль которых будут располагаться ряды автоматов, управляемых той же самой вычислительной системой. Я думаю, что эти автоматы будут скорее напоминать современные станки, а не пальцы руки, хотя они станут не такими громоздкими и будут существенно использовать обратную связь от различных датчиков.

СЛЕДУЮЩИЙ ПРОРЫВ

Думаю, что все мы задаемся вопросом, останутся ли компьютеры такими, какими мы их теперь знаем, или же они претерпят радикальные изменения. При обсуждении этого вопроса хорошо было бы точно выяснить, чего мы уже достигли. Принятие принципа, что процессор в каждый момент времени занимается только одним делом, по крайней мере с точки зрения программиста, сделало программирование концептуально очень простым и проложило путь для последовательного усложнения, уровень за уровнем, чему мы все были свидетелями. Наблюдения за попытками программировать на первых компьютерах, в которых умножения и другие операции выполнялись параллельно, привели меня к мнению, что важность упомянутого принципа трудно переоценить. Что касается аппаратного обеспечения, тот же принцип привел к разработке систем, в которых высокий коэффициент использования аппаратуры поддерживался применительно к самой разной проблематике, т. е. к разработке поистине универсальных компьютеров. Вычислительная машина ЭНИАК, наоборот, содержала уйму аппаратуры, часть которой была предназначена для вычислений, а часть — для программирования, но применительно к среднестатистической задаче только сравнительно небольшая доля этой аппаратуры использовалась в любой конкретный момент времени.

Если произойдут революционные сдвиги, они должны быть сопряжены с внедрением высокой степени параллелизма, который станет возможным благодаря применению интегральных схем. Проблема состоит в том, чтобы добиться достаточно высокого коэффициента использования аппаратуры, потому что без этого параллелизм не принесет нам большей эффективности. Сильно запараллеленные системы зачастую оказываются эффективными только применительно к тем задачам, которые имел в виду проектировщик. Для других задач наблюдается тенденция к падению коэффициента использования аппаратуры до столь низкого уровня, что при длительном счете обычные компьютеры оказываются более эффективными. Я думаю, что для сильно запараллеленных систем нам неизбежно придется принимать более высокую степень специализации, ориентированной на конкретные области прикладных задач, чем принято в настоящее время. Безусловно, важным фактором является абсолютная стоимость интегральных схем, но следует заметить, что значительное снижение такой стоимости оказалось бы благоприятным и для компьютеров с традиционной архитектурой.

Существует проблематика, в которой, по моему мнению, мы должны возлагать особые надежды на высокую степень параллелизма. Это распознавание образов в двух измерениях. Современные компьютеры оказываются удручающе неэффективными при решении таких задач. Здесь я имею в виду не только распознавание рукописных букв. Многие проблемы символьных манипуляций включают в себя значительный элемент распознавания образов, хорошим примером может служить синтаксический анализ. Я не исключаю возможности того, что произойдет большой концептуальный прорыв в области распознавания образов, который революционным образом преобразует всю проблематику машинных вычислений.

РЕЗЮМЕ

Я занимался разными разделами информатики с ее ранних дней по нынешнего времени. Я начинал не с самого начала, потому что первооткрыватели работали не с электронными, а с механическими и электромеханическими устройствами. Тем не менее мы очень обязаны им, и я полагаю, что даже теперь их работы полезно изучать.

Рассматривая ретроспективно, как менялись интересы специалистов по вычислительной технике и программированию и как постоянно расширялся круг пользователей вычислительных систем, нельзя не удивиться способности компьютеров связывать подлинной взаимной заинтересованностью людей с самы-

ми разнообразными исходными целями и побуждениями. Именно этому мы обязаны жизнестойкостью и энергией нашей Ассоциации. Если даже сочтут необходимым изменить ее название, я надеюсь, что сохранятся слова «вычислительная техника» или некий общепонятный синоним. Ибо всех нас связывает не какая-то абстракция наподобие машины Тьюринга или информации, а реальная аппаратура, с которой мы повседневно работаем.