Цифровой компьютер — это машина, которая может решать задачи, исполняя данные ей команды. Последовательность команд, описывающих решение определенной задачи, называется программой. Электронные схемы каждого компьютера могут распознавать и исполнять ограниченный набор простых команд. Все программы перед исполнением должны быть превращены в последовательность таких команд, которые обычно не сложнее, чем, например:

- сложить два числа;
- проверить, не является ли число нулем;
- скопировать блок данных из одной части памяти компьютера в другую.

Эти примитивные команды в совокупности составляют язык, на котором люди могут общаться с компьютером. Такой язык называется машинным. Разработчик при создании нового компьютера должен решить, какие команды следует включить в машинный язык этого компьютера. Это зависит от назначения компьютера и от задач, которые он должен решать. Обычно стараются сделать машинные команды как можно проще, чтобы избежать сложностей при разработке компьютера и снизить затраты на необходимую электронику. *Большинство машинных языков крайне примитивны, из-за чего писать на них и трудно, и утомительно.*

Это простое наблюдение с течением времени привело к построению ряда уровней абстракций, каждая из которых надстраивается над абстракцией более низкого уровня. Именно таким образом можно преодолеть сложности и сделать процесс проектирования систематичным и организованным. Этот подход называется многоуровневой компьютерной организацией.

Существует огромная разница между тем, что удобно людям, и тем, что могут компьютеры. Люди хотят сделать X, но компьютеры могут сделать только Y. Из-за этого возникает проблема.

Эту проблему можно решить двумя способами. Оба способа подразумевают разработку новых команд, более удобных для человека, чем встроенные машинные команды. Эти новые команды в совокупности формируют язык, который назовем Я1. Встроенные машинные команды тоже формируют язык, назовем его Я0. Компьютер может исполнять только программы, написанные на его машинном языке Я0. Два способа решения проблемы отличаются тем, каким образом компьютер будет исполнять программы, написанные на языке Я1 — ведь, в конечном итоге, компьютеру доступен только машинный язык Я0.

Первый способ исполнения программы, написанной на языке Я1, подразумевает замену каждой команды эквивалентным набором команд на языке Я0.

В этом случае компьютер исполняет новую программу, написанную на языке Я0, вместо старой программы, написанной на Я1. Эта технология называется трансляцией.

Второй способ заключается в создании на языке Я0 программы, получающей в качестве входных данных программы, написанные на языке Я1. При этом каждая команда языка Я1 обрабатывается поочередно, после чего сразу исполняется эквивалентный ей набор команд языка Я0. Эта технология не требует составления новой программы на Я0. Она называется интерпретацией, а программа, которая осуществляет интерпретацию, называется интерпретатором.

Между **трансляцией** и **интерпретацией** много общего. В обоих случаях компьютер в конечном итоге исполняет набор команд на языке Я0, эквивалентных командам Я1. Отличие лишь в том, что при трансляции вся программа Я1 переделывается в программу Я0, программа Я1 отбрасывается, а новая программа на Я0 загружается в память компьютера и затем исполняется. Во время выполнения сгенерированная программа на Я0 управляет работой компьютера.

При **интерпретации** каждая команда программы на Я1 перекодируется в Я0 и сразу же исполняется. Транслированная программа при этом не создается.

Работой компьютера управляет **интерпретатор**, для которого программа на Я1 есть не что иное, как «сырые» входные данные. Оба подхода широко используются как вместе, так и по отдельности.

Впрочем, чем мыслить в терминах трансляции и интерпретации, гораздо проще представить себе существование гипотетического компьютера или виртуальной машины, для которой машинным языком является язык Я1. Назовем такую виртуальную машину М1, а виртуальную машину для работы с языком Я0 — М0. Если бы такую машину М1 можно было бы сконструировать без больших денежных затрат, язык Я0, да и машина, которая исполняет программы на языке Я0, были бы не нужны. Можно было бы просто писать программы на языке Я1, а компьютер сразу бы их исполнял. Даже с учетом того, что создать виртуальную машину, возможно, не удастся (из-за чрезмерной дороговизны или трудностей разработки), люди вполне могут писать ориентированные на нее программы. Эти программы будут транслироваться или интерпретироваться программой, написанной на языке Я0, а сама она могла бы исполняться существующим компьютером. Другими словами, можно писать программы для виртуальных машин так, как будто эти машины реально существуют.

Трансляция и **интерпретация** целесообразны лишь в том случае, если языки Я0 и Я1 не слишком отличаются друг от друга, т.е. оперируют близкими понятиями. Часто это значит, что язык Я1 хотя и лучше, чем Я0, но все же далек от идеала. Возможно, это несколько обескураживает в свете первоначальной цели создания языка Я1 — освободить программиста от бремени написания программ на языке, понятным компьютеру, но малоприспособленном для человека. Однако ситуация не так безнадежна.

Очевидное решение проблемы — создание еще одного набора команд, которые в большей степени ориентированы на человека и в меньшей степени на компьютер, чем Я1. Этот третий набор команд также формирует язык,

который мы будем называть Я2, а соответствующую виртуальную машину — М2. Человек может писать программы на языке Я2, как будто виртуальная машина для работы с машинным языком Я2 действительно существует. Такие программы могут либо транслироваться на язык Я1, либо исполняться интерпретатором, написанным на языке Я1.

Изобретение целого ряда языков, каждый из которых более удобен для человека, чем предыдущий, может продолжаться до тех пор, пока мы не дойдем до подходящего нам языка. Каждый такой язык использует своего предшественника как основу, поэтому мы можем рассматривать компьютер в виде ряда уровней, изображенных на рис. 1.1. Язык, находящийся в самом низу иерархической структуры — самый примитивный, а тот, что расположен на ее вершине — самый сложный.

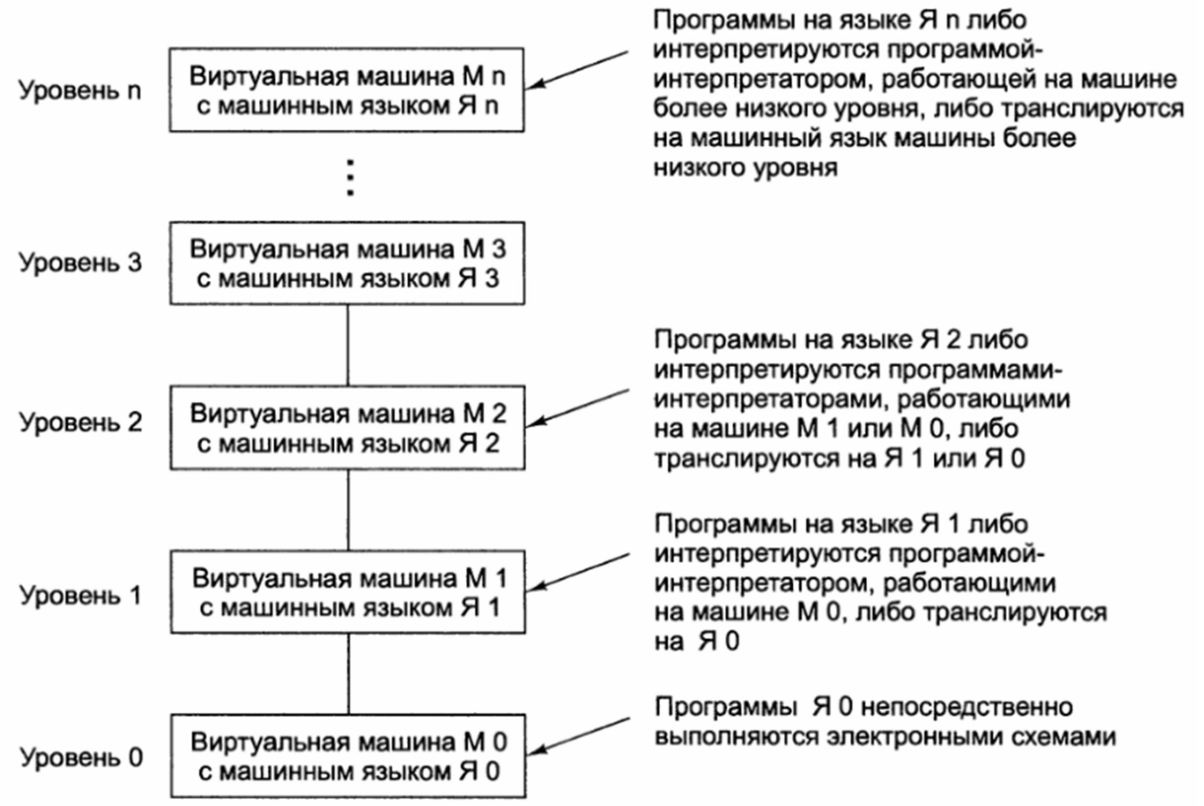

Рис 1.1.1 Многоуровневая машина

Большинство современных компьютеров состоит из двух и более уровней.

Существуют компьютеры с шестью уровнями (рис. 1.1.2). Уровень 0 — это аппаратное обеспечение машины. Электронные схемы на уровне 1 выполняют машинно-зависимые программы. Ради полноты нужно упомянуть о существовании еще одного уровня, который расположен ниже нулевого. Этот уровень не показан на рис. 1.1.2, так как он попадает в сферу электронной техники и называется уровнем *физических устройств.*

рис. 1.1.2 Шестиуровневый компьютер. Способ поддержки каждого уровня показан под ним, в скобках дано название соответствующего программного обеспечения

На самом нижнем цифровом логическом уровне, объекты называются вентилями. Хотя вентили состоят из аналоговых компонентов, таких как транзисторы, они могут быть точно смоделированы как цифровые устройства. У каждого вентиля есть один или несколько цифровых входов (сигналов, представляющих 0 или 1). Вентиль вычисляет простые функции этих сигналов, такие как *И* или *ИЛИ* Каждый вентиль формируется из нескольких транзисторов. Несколько вентилей формируют 1 бит памяти, который может содержать 0 или 1.

Биты памяти, объединенные в группы, например, по 16, 32 или 64, формируют **регистры.** Каждый регистр может содержать одно двоичное число до определенного предела. Из вентилей также может состоять сам компьютер.

Следующий уровень называется уровнем **микроархитектуры.** На этом уровне находятся совокупности 8 или 32 регистров, которые формируют локальную память и схему, называемую АЛУ (арифметико-логическое устройство). АЛУ выполняет простые арифметические операции. Регистры вместе с АЛУ формируют тракт данных, по которому поступают данные. Тракт данных работает следующим образом. Выбирается один или два регистра, АЛУ производит над ними какую-либо операцию, например сложения, после чего результат вновь помещается в один из этих регистров.

У первых цифровых компьютеров 40-х годов было только два уровня: уровень архитектуры набора команд, на котором осуществлялось программирование, и цифровой логический уровень, на котором программы исполнялись. Схемы цифрового логического уровня были ненадежны, сложны для производства и понимания.

В 1951 году Морис Уилкс (Maurice Wilkes), исследователь Кембриджского университета, предложил идею трехуровневого компьютера, призванную радикально упростить аппаратное обеспечение, а следовательно, сократить количество (ненадежных) электронных ламп. Эта машина должна была иметь встроенный неизменяемый интерпретатор (микропрограмму), функция которого заключалась в исполнении программ уровня ISA посредством интерпретации.

Так как аппаратное обеспечение должно было теперь вместо программ уровня ISA исполнять только микропрограммы с ограниченным набором команд, требовалось меньшее количество электронных схем. Поскольку электронные схемы тогда делались из электронных ламп, данное упрощение призвано было сократить количество ламп и, следовательно, повысить надежность (которая в то время выражалась числом поломок за день).

В 50-е годы было построено несколько трехуровневых машин. В 60-х годах число таких машин значительно увеличилось. К 70-м годам идея о том, что написанная программа сначала должна интерпретироваться микропрограммой, а не исполняться непосредственно электроникой, стала преобладающей. В наши дни она используется всеми современными компьютерами.

На некоторых компьютерах работа тракта данных контролируется особой программой, которая называется *макропрограммой.* На других машинах тракт данных контролируется аппаратными средствами.

На компьютерах, где тракт данных контролируется программным обеспечением, микропрограмма — это *интерпретатор* для команд на уровне 2. Микропрограмма вызывает команды из памяти и выполняет их одну за другой, используя при этом тракт данных. Например, при выполнении команды *ADD* она вызывается из памяти, ее операнды помещаются в регистры, АЛУ вычисляет сумму, а затем результат переправляется обратно. На компьютере с аппаратным контролем тракта данных происходит такая же процедура, но при этом нет программы, интерпретирующей команды уровня 2.

Уровень 2 мы будем называть уровнем **архитектуры набора команд.** Каждый производитель публикует руководство для компьютеров, которые он продает, под названием «Руководство по машинному языку X», «Принципы работы компьютера Y» и т. п. Подобное руководство содержит информацию именно об этом уровне. Описываемый в нем набор машинных команд в действительности выполняется микропрограммой-интерпретатором или аппаратным обеспечением.

Если производитель поставляет два интерпретатора для одной машины, он должен издать два руководства по машинному языку, отдельно для каждого интерпретатора.

С 1970 года, когда получило развитие микропрограммирование, производители осознали, что теперь новые машинные команды можно добавлять простым расширением микропрограммы. Иначе говоря, они могли добавлять «аппаратное обеспечение» (новые команды) путем программирования. Это открытие буквально привело к взрыву в производстве наборов машинных команд, поскольку производители начали конкурировать друг с другом — каждый старался, чтобы его набор команд был больше и лучше, чем у других. Многие команды не представляли особой ценности, поскольку те же задачи можно было легко решить, используя уже существующие команды, но обычно они выполнялись немного быстрее. Например, во многих компьютерах использовалась команда INC (INCrement), которая прибавляла к числу единицу. Тогда уже существовала общая команда сложения ADD , и не было необходимости вводить новую команду, прибавляющую к числу единицу. Тем не менее команда INC работала немного быстрее, чем ADD , поэтому ее также включили в набор команд.

Многие команды добавлялись в микропрограмму по той же причине. Среди них можно назвать

команды для: умножения и деления целых чисел;

арифметических действий над числами с плавающей точкой;

вызова и прекращения действия процедур;

ускорения циклов;

работы с символьными строками.

В 60-х–70-х годах количество микропрограмм значительно увеличилось. Однако они работали все медленнее и медленнее, поскольку занимали все больше места.

В конце концов исследователи осознали, что отказ от микропрограмм резко сократит количество команд, и компьютеры станут работать быстрее. Таким образом, компьютеры вернулись к тому состоянию, в котором они находились до изобретения микропрограммирования.

Впрочем, нельзя сказать, что эта ветвь привела в тупик. Современные процессоры продолжают использовать микропрограммы для преобразования сложных команд во внутренний микрокод, который может напрямую выполняться на оптимизированных аппаратных компонентах.

Следующий уровень обычно является **гибридным.** Большинство команд в его языке есть также и на уровне архитектуры набора команд (команды, имеющиеся на одном из уровней, вполне могут быть представлены и на других уровнях).

У этого уровня есть некоторые дополнительные особенности: новый набор команд, другая организация памяти, способность выполнять две и более программы одновременно и некоторые другие. При построении уровня 3 возможно больше вариантов, чем при построении уровней 1 и 2.

Новые средства, появившиеся на уровне 3, выполняются интерпретатором, который работает на втором уровне. Этот интерпретатор был когда-то назван ***операционной системой***. Команды уровня 3, идентичные командам уровня 2, выполняются микропрограммой или аппаратным обеспечением, но не операционной системой. Другими словами, одна часть команд уровня 3 интерпретируется операционной системой, а другая часть — микропрограммой. Вот почему этот уровень считается гибридным.

Операционная система была придумана для того, чтобы автоматизировать работу оператора (отсюда и название), она стала первым шагом в развитии новой виртуальной машины. И хотя этот уровень состоял изначально всего из двух команд, он стал первым шагом в развитии виртуальных машин.

В последующие годы операционные системы все больше и больше усложнялись. К уровню архитектуры набора команд добавлялись новые команды, приспособления и функции, из которых в конечном итоге сформировался

Между уровнями 3 и 4 есть существенная разница. Нижние три уровня задуманы не для того, чтобы с ними работал обычный программист. Они изначально ориентированы на интерпретаторы и трансляторы, поддерживающие более высокие уровни. Эти трансляторы и интерпретаторы составляются так называемыми системными программистами, которые специализируются на разработке новых виртуальных машин. Уровни с четвертого и выше предназначены для прикладных программистов, решающих конкретные задачи.

Еще одно изменение, появившееся на уровне 4, — механизм поддержки более высоких уровней Уровни 2 и 3 обычно интерпретируются, а уровни 4, 5 и выше обычно, хотя и не всегда, транслируются.

Другое различие между уровнями 1, 2, 3 и уровнями 4, 5 и выше — особенность языка. Машинные языки уровней 1, 2 и 3 — *цифровые.* Программы, написанные на этих языках, состоят из длинных рядов цифр, которые воспринимаются компьютерами, но малопонятны для людей. Начиная с уровня 4, языки содержат слова и сокращения, понятные человеку.

Уровень 4 представляет собой символическую форму одного из языков более низкого уровня. На этом уровне можно писать программы в приемлемой для человека форме. Эти программы сначала транслируются на язык уровня 1, 2 или 3, а затем интерпретируются соответствующей виртуальной или фактически существующей машиной. Программа, которая выполняет трансляцию, называется **ассемблером.**

Уровень 5 обычно состоит из языков, разработанных для прикладных программистов. Такие языки называются языками высокого уровня. Существуют сотни языков высокого уровня. Наиболее известные среди них — *C, C++, Java, Python* и *Perl.* Программы, написанные на этих языках, обычно транслируются на уровень 3 или 4. Трансляторы, которые

обрабатывают эти программы, называются **компиляторами.** Иногда также имеет место интерпретация.

В некоторых случаях уровень 5 состоит из интерпретатора для конкретной прикладной области Он предусматривает данные и операции для решения задач в этой области, выраженные при помощи специальной терминологии.

Таким образом, компьютер проектируется как иерархическая структура уровней, которые надстраиваются друг над другом. Каждый уровень представляет собой определенную абстракцию различных объектов и операций.

Программы, написанные на машинном языке (уровень 1) см рис. 1.1.2, могут сразу без применения интерпретаторов и трансляторов исполняться электронными схемами компьютера (уровень 0). Эти электронные схемы вместе с памятью и средствами ввода-вывода формируют аппаратное обеспечение компьютера. **Аппаратное обеспечение состоит** из материальных объектов — интегральных схем, печатных плат, кабелей, источников электропитания, модулей памяти и принтеров. Абстрактные понятия, алгоритмы и команды к аппаратному обеспечению не относятся.

Программное обеспечение, напротив, состоит из алгоритмов (подробных последовательностей команд, которые описывают решение некоторой задачи) и их компьютерных представлений, то есть программ. Программы могут храниться на жестком диске, гибком диске, компакт-диске или других носителях, но это не так уж важно; в сущности, программное обеспечение — это набор команд, составляющих программы, а не физические носители, на которых эти программы записаны.

В самых первых компьютерах граница между аппаратным и программным обеспечением была очевидна. Однако со временем произошло значительное размывание этой границы, в первую очередь благодаря тому, что в процессе развития компьютеров уровни добавлялись, убирались и сливались между собой. В настоящее время очень сложно отделить их друг от друга.

Таким образом, можно сказать, что ***Аппаратное и программное обеспечение логически эквивалентно***.

Любая операция, исполняемая программным обеспечением, может быть реализована аппаратным обеспечением. Как говорится «***Аппаратное обеспечение — это всего лишь окаменевшее программное обеспечение***».

Обратное тоже верно: любая команда, исполняемая аппаратным обеспечением, может быть смоделирована программно. Решение о разделении функций аппаратного и программного обеспечения основано на таких факторах, как стоимость, быстродействие, надежность, частота ожидаемых изменений. Незыблемых правил, требующих, чтобы операция X была реализована в аппаратном обеспечении, а операция Y непременно программировалась, очень мало. Эти решения меняются в зависимости от тенденций экономического и технологического развития.

Важно запомнить, что компьютер проектируется как иерархическая структура уровней, которые надстраиваются друг над другом.

Каждый уровень представляет собой абстракцию некоторых объектов и операций. Рассматривая и анализируя строение компьютера подобным образом, мы можем не принимать во внимание лишние подробности и, таким образом, сделать сложный предмет более простым для понимания.

Набор типов данных, операций и характеристик каждого отдельно взятого уровня называется архитектурой. Архитектура связана с аспектами, видимыми пользователю этого уровня. Например, сведения о том, сколько памяти можно использовать при написании программы, — часть архитектуры. Аспекты реализации (например, технология, применяемая при реализации памяти) не являются частью архитектуры. Изучая методы проектирования программных элементов компьютерной системы, мы изучаем компьютерную архитектуру. На практике термины «компьютерная архитектура» и «компьютерная организация» употребляются как синонимы.

Граница между аппаратным и программным обеспечением постоянно смещается. Сегодняшнее программное обеспечение может быть завтрашним аппаратным обеспечением, и наоборот. Более того, также обстоит дело и с уровнями — между ними нет четких границ. Для программиста не важно, как на самом деле исполняется команда (за исключением, может быть, скорости исполнения). Программист, работающий на уровне архитектуры набора команд, может использовать команду умножения, как будто это аппаратная команда, и даже не задумываться об этом. То, что для одного человека — программное обеспечение, для другого — аппаратное.

Основные вычислительные задачи начала XX в.

Астрономические расчеты и навигация

Астрономия и навигация ставили вычислительные задачи с давних времен. Со временем интенсивность и значимость морских сообщений существенно возросла. Тот же Бэббидж, создавая «разностный вычислитель», получил финансирование именно под составление мореходных таблиц (обычно это таблицы умножения, логарифмов, синусов, косинусов, а также всевозможные таблицы результатов астрономических и навигационных измерений и наблюдений).

Расширился и круг интересов астрономии, одним из вычислительных достижений которой стало предсказание карликовой планеты Плутона по возмущениям орбиты Нептуна и Урана (конец XIX в.).

Кораблестроение

Наряду с астрономией и навигацией, начиная с XVIII в. Особенно нуждались в точных расчетах корабельные науки. Они требовали расчета устойчивости и усилий, возникающих в конструкции корабля при качке, определения «ходкости» и устойчивости корабля под парусами, расчета нагрузок на конструкции корабля от отдачи орудий и т. д. Еще в конце XVIII

в. было создано много алгоритмов для решения практических задач, но вычисления по этим алгоритмам были чрезвычайно трудоемки.

Рубеж же XIX и XX веков дополнительно ознаменовался «гонкой морских вооружений» (приостановленной лишь Вашингтонскими соглашениями в 1922 г.), когда новейшие броненосцы в момент спуска на воду уже оказывались морально устаревшими. Именно для расчета корпусов судов А.Н. Крылов в 1904 г. создал один из первых аналоговых компьютеров

Статистика, экономика и бухучет

Первоначально статистика вращалась вокруг потребности государства знать демографические и экономические параметры общества. К XX веку объем дисциплины стал расширяться, и в середине века статистические расчеты уже использовались, например, при создании ядерной бомбы (метод Монте-Карло).

На начало XX века приходятся и попытки математического описания экономики – труды основоположников (Смита, Рикардо) анализируются математическими методами, создаются новые экономические теории (Кейнс). Быстрый рост монополий и увеличение вмешательства государств в экономику ставят задачу ведения бухгалтерского и налогового учета в огромных масштабах.

Ядерная физика

Создание США ядерной бомбы потребовало проведения значительного объема математических расчетов из-за неясности относительно оптимального способа подрыва бомбы и характера её поражающего фактора (взрывной волны), осложненной нехваткой радиоактивных материалов и отсутствием времени для многократных испытаний. После успешного применения бомбы США задача СССР по созданию собственного оружия существенно упростилась, поскольку остались преимущественно инженерные задачи. Но последующая разработка водородной бомбы вновь потребовала полноценного математического моделирования.

Баллистические расчеты

Попытки описания модели полета снаряда предпринимались со времен изобретения пушек, но из-за неполноты моделей и сложности расчетов баллистические таблицы, позволяющие выбрать угол вертикальной наводки орудия при заданных условиях, долгое время составлялись исключительно путем испытания орудий на полигонах.

К концу XIX века развитие аэродинамики позволило найти достаточно точное математическое описание сил, действующих на тело, движущееся с большой скоростью в воздухе, а также был разработан численный метод интегрирования дифференциальных уравнений, позволявший с заданной точностью решать баллистические уравнения.

Оставалась только проблема объема самих вычислений. Для расчета одной траектории было необходимо выполнить минимум 750 операций умножения, на что квалифицированный специалист с арифмометром затрачивал около 3 дней, а для каждой комбинации орудия и снаряда требовалось 2–4 тыс. таких расчетов.

Криптография

В XIX и начале XX веков несколько факторов способствовали развитию криптографии. Первым фактором были детективные истории, такие как «Золотой жук» Эдгара По или «Пляшущие человечки» Конан Дойля, в которых фигурировали закодированные сообщения и которые волновали воображение многих читателей. Вторым фактором явилось изобретение телеграфа и азбуки Морзе. Азбука Морзе была первым двоичным представлением алфавита, которое получило широкое распространение. Однако в то время сложные шифры применялись не часто, так как требовали много времени и сил для кодирования и декодирования. Также большое влияние оказывали быстрорастущие фондовые биржи, банки и монополии, создававшие спрос на способы конфиденциальной передачи сведений в больших объемах.

В первую мировую войну в ряде стран были разработаны роторные шифровальные машины, которые позволяют легко кодировать и декодировать текст, используя сложный шифр. Сама по себе идея роторного шифрования известна со времен античности, когда был изобретен шифр Цезаря, основанный на замене одной буквы другой буквой по определенному правилу. Одной из первых практически используемых машин, стала немецкая Enigma, разработанная в 1917 году. В модернизированном варианте она была принята на вооружение в Германии (общий тираж около 100 тыс. экз.), а в разных модификациях была доступна для коммерческого применения.

У американцев во время Второй мировой войны была собственная очень мощная версия 15-дискового шифратора Sigaba. Всего было выпущено до 10 тыс. устройств Sigaba, они продержались на вооружении до конца 50-х годов. В Великобритании производился шифратор Typex, собственный аналог Enigma. Работы по созданию механизированных шифраторов велись и в СССР, созданная аппаратура обеспечивала шифрование, считавшееся абсолютно надежным, но выпускалась несравненно малыми тиражами (на момент начала ВОВ на вооружении стояло около 250 комплектов).

Дальние линии электропередач

В дальних линиях электропередач начинают сказываться индуктивность и емкость провода, ограничивая пропускную способность линий и вызывая параметрическую неустойчивость – самовозбуждение генераторов электростанций. Постройка линий стала требовать предварительного расчета режимов работы сети, особенно при включении их в единую энергосистему или подключении масштабных регионов – потребителей энергии.

Поколения компьютеров

Единого мнения относительно того, что считать «первым» компьютером, нет. «Первыми» могут считаться больше десятка моделей:

• Ткацкий станок Жаккарда (1801) – первая не счетная программируемая машина;

• Analytical Engine Бэббиджа (1834) – первый программируемый компьютер (спроектированный);

• Машина Тьюринга (1936) – первая теоретическая модель вычислительной машины, работающей по алгоритму;

• Z3 Цузе (1941) – первый программируемый цифровой компьютер (работающий);

• ABC Атанасова и Берри (1942) – первый специализированный электронный цифровой компьютер;

• Mischgerät Хельцера (1943) – первый бортовой компьютер;

• Colossus Флауэрса (1943) – первый электронный компьютер (в Великобритании);

• Harvard Mark I Айкена (1944) – первый программируемый компьютер (в США);

• ENIAC Эккерта и Мочли (1946) – первый электронный программируемый компьютер (США);

• Manchester SSEM Baby Килбурна (1948) – первый компьютер с хранимой программой (Великобритания);

• BINAC Эккерта и Мочли (1949) – первый компьютер с хранимой программой (в США);

• МЭСМ Лебедева (1950) – первый компьютер с хранимой программой (в СССР и континентальной Европе).

Причин путаницы в приоритете и датах создания компьютеров сразу несколько. Во-первых, какой момент считать датой создания компьютера: когда возникла идея, когда она была опубликована, когда что-то из задуманного начало работать, или когда компьютер был сдан в опытную эксплуатацию? Во-вторых, о каком компьютере идет речь: аналоговом или цифровом, (электро)механическом или электронном,

специализированном или универсальном? В-третьих, что же вообще понимать под компьютером? Должен ли он отвечать определенным критериям и где проходит грань между изощренным калькулятором, вычислительной машиной и компьютером?

Эта путаница на уровне названий заметна во многих странах. В СССР долгое время употребляли термины «электронная счетная машина», «электронная вычислительная машина» (ЭВМ), а с начала 1980-х годов их вытесняет термин «компьютер». В 1930–1940-е годы слово «computer» обычно означало человека, производящего вычисления, а также любой тип машины, механизирующей вычисления.

В фундаментальной работе Алана Тьюринга «О вычислимых числах» (1936) слово «компьютер» используется исключительно для обозначения человека, а в 1950 г. Тьюринг уже употребляет для ясности термины human computer и digital computer. После 1945 г. термин «компьютер» стал употребляться все чаще, причем практически всегда в смысле автоматического электронного цифрового компьютера с внутренней памятью для программ (automatic electronic digital computer with internal program storage).

«Первым» компьютером было решено считать ENIAC, а вычислительные машины его эпохи позднее были отнесены к первому поколению компьютеров.

По этапам создания и используемой элементной базе ЭВМ условно делятся на поколения:

Первое поколение, 40-е - 50-е годы; ЭВМ на электронных вакуумных лампах.

Второе поколение, 60-е годы; ЭВМ на дискретных полупроводниковых приборах (транзисторах).

Третье поколение, 70-е годы; ЭВМ на полупроводниковых интегральных схемах с малой и средней степенью интеграции (сотни – тысячи транзисторов в одном корпусе).

Четвертое поколение, 80-е годы; ЭВМ на больших и сверхбольших интегральных схемах – микропроцессорах (десятки тысяч – миллионы транзисторов в одном в одном корпусе).

Пятое поколение, 90-е годы; ЭВМ со многими десятками параллельно работающих микропроцессоров, позволяющих строить эффективные системы обработки знаний; ЭВМ на сверхсложных микропроцессорах с параллельно-векторной структурой, одновременно выполняющих десятки последовательных команд программы;

Шестое и последующие поколения; оптоэлектронные ЭВМ с массовым параллелизмом и нейтронной структурой – с распределенной сетью большого числа (десятки тысяч) несложных микропроцессоров, моделирующих архитектуру нейтронных биологических систем.

Каждое следующее поколение ЭВМ имеет по сравнению с предыдущими существенно лучшие характеристики. Так, производительность ЭВМ и емкость всех запоминающих устройств увеличивается, как правило, больше чем на порядок.

Проект IBM System/360 (1964)

В 1961 г. в компании IBM, в то время разрабатывающей два разных типа машин, сочли недостаточно революционной новую модель компьютера для научного применения (8000-ю серию), и раскритиковали его несовместимость с машинами общего (коммерческого) назначения.

Работу по созданию «новой линии продуктов» (New Product Line), объединившую оба направления, возглавил Б. Эванс. Для выработки концепции NPL был создан комитет SPREAD (Systems Programming Research and Development), изложивший семь основных принципов:

• центральный процессор должен с равным успехом использоваться для научных и деловых вычислений;

• все члены будущего семейства должны быть способны работать с одним и тем же набором периферийных устройств;

• для центрального процессора была выбрана гибридная технология, допускавшая использование микросхем и навесной монтаж дискретных элементов (к тому времени уже существовали микросхемы, но они еще не обладали достаточной надежностью);

• в научных и бизнес-вычислениях должен использоваться один и тот же язык программирования высокого уровня;

• между всеми членами семейства должна сохраняться программная совместимость;

• адресация в семействе должна обеспечивать доступ к 16 млн. символов, а в перспективе – к 2 млрд.;

• минимальной единицей представления данных будет 8-битовый байт.

В дальнейшем проект был переименован из NPL в System/360, что должно было означать способность решения любых задач. По своему размаху проект создания System/360 один из крупнейших коммерческих в истории, сопоставимый с проектами полета на Луну и освоения западносибирских нефтегазовых месторождений. В течение короткого промежутка времени в один проект было инвестировано свыше 5 млрд. долл. (по курсу начала 2000-х гг. это свыше 30 млрд. долл.). Для IBM это была рискованная игра, т.к. за несколько лет было заново создано практически все: архитектура, элементная база и системное программное обеспечение.

Проект повлиял и на развитие вычислительной техники в СССР, где был «клонирован» как ЕС ЭВМ.

БЭСМ-6 (1967)

В развитии отечественной вычислительной техники особое место занимает ЭВМ БЭСМ-6 (Быстродействующая Электронно-Счетная Машина). БЭСМ-6 была разработана под руководством С.А. Лебедева, производство начато в 1967 г. и продолжалось до 1987 г. Она была задумана как ЭВМ для расчетов в самых различных областях науки и техники и для оснащения крупных вычислительных центров. Формально ее относили к ЭВМ 2-го поколения, так как ее элементная база выполнена на дискретных элементах. Но по всем прочим признакам – иерархической системе памяти с постоянной организацией, наличию виртуальной памяти, большому числу каналов, обслуживающих периферийные устройства и внешнюю память, наличию эффективных операций с плавающей запятой, наличие сверхбыстрого ЗУ, системы индексации команд, системы прерывания, наличию операционной системы и т.д. – БЭСМ-6 являлась вычислительной системой 3-го поколения. В течение нескольких лет она была самой высокопроизводительной ЭВМ в Европе. «Особое» место БЭСМ-6 занимает и потому, что в 1967 году было принято решение руководства СССР о копировании System/360, и БЭСМ-6 стала последним оригинальным компьютером для массового применения.

В системе команд БЭСМ-6 предусмотрено 50 команд, разделяемых на шесть групп по назначению, и макрокоманды. Код макрокоманды вызывает прерывание, и операционная система передает управление программе с номером данной макрокоманды.

Ряд необычных схемных решений в арифметическом устройстве (АУ) обеспечил возможность при тактовой частоте 10 МГц получить высокую скорость выполнения операций: сложение – 1,1 мкс, умножение – 1,9 мкс, деление – 4,9 мкс, прочие операции – 0,5 мкс. Это позволяет оценить среднее быстродействие АУ в 1 млн. оп/с. Чтобы быстродействие АУ не

ограничивалась низкой скоростью записи и считывания данных и команд из устройств памяти, в БЭСМ-6 были применены 16 полноразрядных регистров сверхбыстрой памяти, что существенно ускорило ход вычислений. Отбор чисел и команд в них велся автоматически из наиболее часто повторяющихся при выполнении программы адресов.

БЭСМ-6 применялась до начала 90-х гг. С её помощью обрабатывалась телеметрическая информация и о полете «Союз-Апплон» в 1975 г., и о полете «Бурана» в 1988 г. В начале 80-х выпускался и вариант БЭСМ на базе интегральных схем, что подтверждает прогрессивность архитектуры комплекса.