

Глава 1. Понятие об архитектуре ЭВМ

Эта книга называется "Введение в архитектуру ЭВМ и системы программирования" и сначала нужно определить, что надо понимать под *архитектурой* компьютера. На бытовом уровне термин "архитектура" у большинства людей прочно ассоциируется с различными зданиями и другими инженерными сооружениями. Так, можно говорить об архитектуре готического собора, Эйфелевой башни или оперного театра. В других областях этот термин применяется достаточно редко, например, было выражение "архитектор перестройки". Для компьютеров понятие "архитектура ЭВМ" уже прочно устоялось и широко используется, начиная примерно с 60-х годов прошлого века.

Заметим сначала, что в общем смысле архитектура существует у любого достаточно сложного объекта, состоящего из отдельных, взаимодействующих между собой частей (компонентов). Так, определяя на бытовом уровне понятие "архитектура ЭВМ", обычно говорят, что архитектура – это все компоненты компьютера, их устройство, выполняемые ими функции, а также взаимосвязи между этими компонентами. Сейчас надо разобраться, почему такое поверхностное определение архитектуры не будет нас удовлетворять.¹

Дело в том, что понятие *архитектуры* чего-либо существует не само по себе, а только в паре с другим понятием. Такая ситуация встречается достаточно часто. Известно, например, что понятие *алгоритм* неразрывно связано с понятием *исполнитель* алгоритма. При этом одна и та же запись для одного исполнителя будет алгоритмом, а для другого – нет (например, если этот другой исполнитель не умеет выполнять некоторые предписания в записи алгоритма).

Так и в рассматриваемом случае понятие архитектуры неразрывно связано с тем человеком (или теми людьми), которые изучают или рассматривают эту архитектуру. Ясно, что для разных людей архитектура одного и того же объекта может выглядеть совершенно по-разному. Так, например, обычный жилец многоэтажного дома не без основания полагает, что этот дом состоит из фундамента, стен и крыши, имеет этажи, на каждом этаже есть квартиры, присутствует лестница, лифт, в квартирах есть комнаты, окна, двери и т.д.

Совсем по-другому видит архитектуру этого же дома инженер, ответственный за его эксплуатацию. Он, например, также знает, что некоторые перегородки между комнатами можно убрать при перепланировке квартиры, а другие перегородки являются *несущими*, если их убрать – дом рухнет. Инженер знает, где внутри стен проходят электрические провода, трубы водяного отопления, как обеспечивается противопожарная безопасность, каким образом к дому подводятся инженерные коммуникации и многое другое.

Отсюда можно сделать вывод, что, изучая какой-либо объект, часто бывает удобно выделить различные **уровни** рассмотрения архитектуры этого объекта. Обычно удобно выделить три таких уровня, их часто называют *внешним*, *концептуальным* и *внутренним*.² В качестве примера давайте рассмотрим архитектуру какого-нибудь всем хорошо известного объекта, например, легкового автомобиля, на этих трёх уровнях.

1. **Внешний уровень.** На этом уровне видит архитектуру легкового автомобиля обычный пассажир. Он знает, что машина имеет колёса, кузов, сиденья, руль, мотор и другие части. Он понимает, что для работы автомобиля в него надо обязательно заливать бензин, знает назначение дворников на ветровом стекле, ремней безопасности и т.д. И этого ему вполне достаточно, чтобы успешно пользоваться машиной, главное – правильно назвать водителю нужный адрес 😊.
2. **Концептуальный уровень.** Примерно на этом уровне видит архитектуру машины её водитель. В отличие от пассажира он знает, что в *его* автомобиль нужно заливать вовсе не бензин, а дизельное топливо (а для электромобиля нужно вообще "заливать" только электричество).

¹ Совокупность компонент компьютера, а также взаимосвязи между этими компонентами в научной литературе иногда называют *структурой* компьютера.

² Числа три и четыре являются особыми в человеческом мышлении, достаточно вспомнить "у отца было три сына", "загадать три желания", четыре стороны света, четыре сезона года и т.д. По-видимому, это связано с тем, что при рассуждениях для человека два – это ещё "слишком мало", а пять – уже "слишком много" для первичного деления сложного объекта на части. В области компьютеров и программного обеспечения такие три уровня можно использовать, например, при описании баз данных. Структура хранящихся данных (так называемая схема данных) может описываться и рассматриваться на внешнем, концептуальном и внутреннем уровнях [12].

Кроме того, в автомобиль необходимо ещё заливать масло определенной марки и специальную воду для стеклоочистителей. Водитель знает назначение всех органов управления машиной, марку топлива, температуру окружающего воздуха, ниже которой необходимо заливать в машину особый сорт масла и ставить зимнюю резину т.д. Обычно водитель обладает также некоторыми знаниями, позволяющими выполнить несложный ремонт машины. Ясно, что наш водитель видит архитектуру своего автомобиля совсем иначе, нежели обычный пассажир.

3. **Внутренний уровень.** На этом уровне автомобиль видит инженер-конструктор, ответственный за его разработку. Он знает марку металла, из которого изготавливаются цилиндры двигателя, зависимость отдаваемой мотором мощности от марки топлива, допустимую нагрузку на отдельные узлы автомобиля, антикоррозионные свойства внешнего корпуса, особенности работы системы безопасности, настройки бортового компьютера и многое другое. Ясно, что обычный водитель машины, а тем более её пассажир, в своей жизни вполне может обойтись без всех этих специальных знаний.

Не надо думать, что один уровень видения архитектуры "хороший", а другой – "плохой". Каждый из них необходим и достаточен для конкретного применения рассматриваемого объекта.¹ Знать объект на более глубоком уровне архитектуры часто бывает даже вредно, так как получить эти знания обычно достаточно трудно, и все усилия пропадут, если в дальнейшем эти знания не понадобятся. Хотя и говорят, что знания лишними не бывают, но лишними часто могут оказаться усилия, потраченные на приобретение этих знаний.

Для уровня университетского образования необходимо, чтобы его выпускники, изучая какой-либо объект, достаточно ясно представляли себе, на каком уровне архитектуры они его рассматривают и достаточен ли этот уровень для практической работы с этим объектом. При необходимости, разумеется, надо перейти на более глубокий уровень рассмотрения изучаемого объекта.

Теперь надо перейти ближе к предмету этой книги – архитектуре компьютеров. Все люди, которые, так или иначе, используют компьютеры в своей деятельности и имеют понятие об их архитектуре, обычно называются **пользователями**. Ясно, что в зависимости от того, на каком уровне они видят архитектуру компьютера, всех пользователей можно, хотя, конечно, и достаточно условно, разделить на уровни или группы (наверное, Вы уже не будете удивлены, что этих групп обычно тоже три). Как правило, выделяют следующие группы пользователей.

1. **Конечные пользователи** (называемые также пользователями-непрограммистами). Для успешного использования компьютеров этим пользователям, как видно из названия, не нужно уметь программировать. Обычно это специалисты в конкретных предметных областях – физики, биологи, лингвисты, финансовые работники и др., либо люди, использующие компьютеры в сфере образования, досуга и развлечений (они имеют дело с обучающими программами, компьютерными играми, навигаторами по сети Интернет и т.д.). В своей работе все они используют компьютер, снабжённый соответствующим, как говорят, *прикладным программным обеспечением* (application software). Это различные базы данных, текстовые редакторы, пакеты прикладных программ, системы автоматического перевода, обучающие, игровые и музыкальные программы и т.п. Таким пользователям достаточно видеть архитектуру компьютеров на *внешнем* уровне, этих людей абсолютное большинство, более 90% от общего числа всех пользователей. Вообще говоря, что бы там себе не воображали пользователи других уровней, компьютеры и программы разрабатываются и выпускаются для нужд именно этих пользователей-непрограммистов. Они *конечные потребители* всей компьютерной индустрии, что и отражается в их названии.

¹ Не следует путать различные *уровни* рассмотрения объекта с рассмотрением этого же объекта с *разных сторон* (с разных точек зрения). Например, можно рассматривать легковой автомобиль, сравнивая его с другими автомобилями, с точки зрения его экономичности, по дизайну и удобству эксплуатации, соотношению цены и качества и т.д. При рассмотрении объекта с некоторой одной стороны остальные стороны могут и совсем не приниматься во внимание. Здесь можно вспомнить известную восточную притчу о трёх слепых, которых подвели к слону и попросили описать его. Один слепой ощупал бок слона и сказал, что он похож на стену, второй, который стоял у ноги, утверждал, что слон похож на колонну, а третий сказал, что слон похож на толстый шланг, так как держался за хобот.

2. **Прикладные программисты.** Как уже ясно из названия, эти пользователи разрабатывают для конечных пользователей прикладное программное обеспечение. В своей работе они чаще всего используют различные языки программирования высокого уровня (Паскаль, Фортран, Си, Java, языки для работы с базами данных и т.д.) и соответствующие *системы программирования* (это понятие достаточно подробно изучается в данной книге). Прикладным программистам достаточно видеть архитектуру компьютеров на *концептуальном* уровне. Можно примерно считать, что прикладных программистов менее 10% от числа всех пользователей. Изучив программирование на языке высокого уровня, Вы должны уже достаточно хорошо представлять себе этот уровень архитектуры компьютера. Попробуйте, используя Ваш программистский опыт, сами сформулировать отличия видения архитектуры ЭВМ на этом уровне, по сравнению с предыдущим уровнем пользователей-непрограммистов.
3. **Системные программисты.** Это самая небольшая (менее одного 1%), но и наиболее квалифицированная группа пользователей, которая видит архитектуру ЭВМ на *внутреннем* уровне. Основная деятельность системных программистов заключается в разработке *системного программного обеспечения*, которое предназначено для автоматизации процесса программирования (это так называемые системы программирования – тот инструмент, с помощью которого программисты разрабатывают, пишут, отлаживают и модифицируют свои программы). Системное программное обеспечение используется и для эффективного управления ресурсами самой ЭВМ, этот комплекс программ называется операционной системой. Заметим, что системы программирования, по аналогии с промышленным производством, можно образно сравнить со *средствами производства* остальных программ. В литературе по архитектуре ЭВМ этот внутренний уровень часто разделяют на два: уровень операционной системы и уровень набора машинных команд, так далее будет сделано и в этой книге.

Разумеется, можно выделить и другие уровни видения архитектуры компьютера, не связанные с его *использованием*. В качестве примера можно указать уровень микроархитектуры, на котором рассматриваются правила выполнения машинных команд, уровень инженера-разработчика аппаратуры компьютера и уровень физика, исследующего новые материалы для построения схем ЭВМ. Эти уровни изучаются на других специальностях, и непосредственно интересовать нас не будут. В этой книге изучаются в основном второй и третий уровни, но иногда, в качестве примеров, совсем немного будут рассматриваться и эти более глубокие уровни видения архитектуры ЭВМ.

Далее укажем те **способы**, с помощью которых в этой книге будет описываться архитектура компьютера. Можно выделить следующие основные способы такого описания.

1. Словесные описания, а также использование чертежей, графиков, рисунков, блок-схем и т.д. Именно таким способом в научной литературе обычно и описывается архитектура ЭВМ для *пользователей* разного уровня.
2. В качестве другого способа описания архитектуры компьютера на *внутреннем* уровне можно с определенным успехом использовать *язык машины* и близкий к нему *язык Ассемблера*. Дело в том, что компьютер является *исполнителем* алгоритма на языке машины и архитектуру компьютера легче понять, если знать язык, на котором записываются эти алгоритмы. В этой книге язык Ассемблера изучается в основном именно для лучшего понимания архитектуры ЭВМ. Для этого понадобится не полный язык Ассемблера, а лишь относительно небольшое подмножество этого языка, достаточное для написания простейших полных программ.
3. Можно проводить описание архитектуры ЭВМ и с помощью *формальных языков*. Из курса по основам теории алгоритмов известно, как важна *формализация* некоторого понятия, что позволяет значительно поднять строгость его описания и устранить различия в понимании этого понятия разными людьми. В основном в настоящее время формальные языки используются для описания архитектуры ЭВМ и её компонентов на *инженерном* уровне, эти языки достаточно сложны, и их изучение выходит за рамки этой книги.¹ В качестве небольшого

¹ В качестве примера можно назвать достаточно широко распространенный язык высокого уровня описания и моделирования электронных устройств Verilog (Verilog Hardware Description Language), с синтаксисом, похожим на язык С, и язык VHDL, с синтаксисом, похожим на язык Ада.

примера, однако, будет дано почти формальное описание архитектуры, но не "настоящего" компьютера, а некоторой *учебной* ЭВМ. Эта ЭВМ будет, с одной стороны, достаточно проста, чтобы её формальное описание не было слишком сложным, а, с другой стороны, она должна быть *универсальной* (т.е. пригодной для реализации любых алгоритмов, для выполнения которых хватает аппаратных ресурсов такого учебного компьютера).

Вы, конечно, уже знаете, что сейчас производятся самые разные компьютеры. Определим теперь, архитектуру **каких** именно ЭВМ будет изучаться в этой книге. Сначала необходимо отметить, что будут рассматриваться только так называемые универсальные цифровые (дискретные) компьютеры, которые в настоящее время составляют абсолютное большинство из всех выпускаемых и используемых ЭВМ. О других способах организации ЭВМ, в частности, о так называемых аналоговых (непрерывных) компьютерах, можно совсем немного (как говорится, "для общего развития") почитать в дополнительной главе в конце этой книги.

Итак, в этой книге изучение архитектуры ЭВМ построено таким образом.

1. Сначала рассматривается архитектура некоторой **абстрактной** вычислительной машины (машины Фон Неймана), на базе которой будут введены основные понятия для последующего изучения архитектуры ЭВМ.
2. Далее будут описаны и изучены специальные **учебные ЭВМ**, которые по своей архитектуре близки к самым первым из выпускавшихся цифровых компьютеров. На основе этих учебных ЭВМ надо понять основные подходы к организации архитектуры компьютеров. Одна из этих учебных машин рассматривается более подробно, чтобы научиться писать для неё простейшие полные программы на языке машины, что поможет Вам лучше понять архитектуру первых компьютеров. Это очень важный момент, так как архитектура первых ЭВМ в той или иной степени лежит в основе конструкций и всех современных компьютеров.
3. Затем достаточно подробно изучается архитектура первой (младшей) модели **конкретного** компьютера фирмы Intel, старшие модели этой ЭВМ наиболее распространены в настоящее время. Кратко упоминаются также изменения в архитектуре для старших моделей этого семейства.
4. В последних главах книги рассматриваются отличительные особенности архитектуры современных компьютеров, а также проводится некоторый достаточно простой **сравнительный анализ** архитектуры основных классов цифровых универсальных ЭВМ.

Вопросы и упражнения

1. Для чего необходимо выделять различные уровни видения архитектуры компьютера?
2. Приведите примеры прикладных программ, которые Вы используете на своем компьютере.
3. Сформулируйте различия в уровнях видения архитектуры компьютера для конечного пользователя и прикладного программиста.