Глава 1. Понятие об архитектуре ЭВМ

Архитектура вычислительной машины (computer architecture). Концептуальная структура вычислительной машины, определяющая проведение обработки информации и включающая методы преобразования информации в данные и принципы взаимодействия технических средств и программного обеспечения 🙄.
ГОСТ 15971-90

Всё должно быть изложено так просто, как только возможно, но не проще.
Альберт Эйнштейн

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, однако, понятие «архитектура ЭВМ» прочно устоялось и широко используется, начиная примерно с 60-х годов прошлого века.

По древнегречески αρχιτεκτων означает искусство проектировать и строить здания. Впервые термин «архитектура» в области вычислительных машин (computer architecture) применил в 1959 году программист фирмы IBM Лайл Джонсон (Lyle R. Johnson). Он сделал это в технической записке «A Description of Stretch» при описании компьютера IBM 7030 Stretch (эта ЭВМ считается прототипом первого суперкомпьютера). Широкой общественности термин «архитектура» стал известен из монографии 1962 года описания этой же ЭВМ под редакцией Фредерика Брукса-младшего (Frederick Phillips Brooks, Jr.), автора знаменитой книги «Мифический человеко-месяц». Вероятно, ему понравился этот термин, именно Брукс определил архитектуру как описание ЭВМ, ориентированное на человека, использующего это описание. Что касается Лайла Джонсона, то впоследствии он стал редактором журнала IBM System Journal и в 1967 году с его лёгкой руки появился и другой важный компьютерный термин – кэш-память (cache memory).

Сначала заметим, 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 в науке достаточно часто. Вам уже должно быть известно, 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, программа, написанная для компьютера на его машинном языке будет для него алгоритмом, однако не будет алгоритмом для ЭВМ с другой системой машинных команд.

Так и в рассматриваемом случае понятие архитектуры неразрывно связано с тем человеком (или теми людьми), которые изучают или рассматривают эту архитектуру. При этом 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 этажах есть квартиры, присутствуют лестницы, лифт, в квартирах есть комнаты, окна, двери и т.д.

---

[1] Совокупность компонент компьютера, а также взаимосвязи между этими компонентами в научной литературе иногда называют *структурой* компьютера. Структура – это отношение между элементами системы [ISO/IEC 2382/1-93].

Совсем по-другому видит архитектуру этого же дома инженер, ответственный за его эксплуатацию. Он, например, 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*.

Числа три и четыре являются особыми в человеческом мышлении, достаточно вспомнить «у отца было три сына», «загадать три желания», у древних римлян в свите богини Венеры было три грации, у нас 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. Кроме того, психологи говорят, что большинство из нас не могут думать о взаимосвязи более, чем о трёх или четырёх вещей одновременно 😟.

В программном обеспечения такие три уровня используются, например, при описании баз данных. Структура хранящихся данных (так называемая *схема* или модель данных) может описываться и рассматриваться на внешнем, концептуальном и внутреннем уровнях [10].

В качестве примера давайте рассмотрим архитектуру какого-нибудь всем хорошо известного объекта, например, легкового автомобиля, на этих трёх уровнях.

1. Внешний уровень (external level). 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 заливать бензин, знает назначение дворников на ветровом стекле, ремней безопасности и т.д. И этого эму вполне достаточно, чтобы успешно пользоваться машиной, главное – правильно назвать водителю нужный адрес 🙂.
2. Концептуальный уровень (conceptual level). Примерно так 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). Кроме того, в автомобиль необходимо ещё заливать масло определенной марки и специальную воду для стеклоочистителей. Водитель знает назначение всех органов управления машиной, марку нужного 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 (internal level). На этом уровне автомобиль, например, видит инженер-конструктор, ответственный за его разработку. Он знает марку металла, из которого изготавливаются цилиндры двигателя, зависимость отдаваемой мотором мощности от октанового числа топлива, допустимую нагрузку на отдельные узлы автомобиля, антикоррозийные свойства 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 ви́дения архитектуры «хороший», а другой – «плохой». Каждый из них необходим и достаточен для конкретного применения рассматриваемого объекта. 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.

Не следует путать различные *уровни* рассмотрения объекта с рассмотрением этого же объекта с *разных сторон* (с разных точек зрения). Например, можно рассматривать конкретный легковой автомобиль, сравнивая его с другими автомобилями, с точки зрения 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.

Для 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 их практической работы с этим объектом. При необходимости, разумеется, надо перейти на более глубокий уровень рассмотрения изучаемого объекта.

Теперь перейдём ближе к предмету этой книги – архитектуре компьютеров.[1] Все люди, которые, так или иначе, используют компьютеры в своей деятельности и имеют понятие об их архитектуре, обычно называются пользователями (пресловутые «юзеры» – users). В зависимости от того, на каком уровне они видят архитектуру компьютера, всех пользователей тоже можно, достаточно условно, разделить на уровни или группы (наверное, Вы уже не будете удивлены, что этих групп тоже три). Как правило, выделяют следующие группы пользователей.

1. Конечные пользователи (end-users, 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 – врачи, биологи, лингвисты, финансовые работники и др. Это также люди, использующие компьютеры в сфере образования, досуга и развлечений (они имеют дело с обучающими программами, социальными сетями, компьютерными играми, навигаторами по сети Интернет и т.д.). В своей работе все они используют компьютер, снабжённый соответствующим, как говорят, прикладным программным обеспечением (ППО – 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 95% от общего числа всех пользователей. И чтобы там себе не думали пользователи других уровней, но компьютеры и программное обеспечение в основном производятся для этих конечных пользователей. Они важные *конечные потребители* всей продукции компьютерной индустрии, что и отражается в их названии.
2. Прикладные программисты (application programmers). Как уже ясно из названия, именно они и разрабатывают для конечных пользователей прикладное программное обеспечение. В своей работе они чаще всего используют языки программирования высокого уровня (Си, Фортран, Java, Python, языки для работы с базами данных, конструкторы Web сайтов и т.д.) и соответствующие *системы программирования* (это понятие будет достаточно подробно изучаться в данной книге). Прикладным программистам достаточно видеть архитектуру компьютеров на *концептуальном* уровне. Можно примерно считать, что прикладных программистов менее 5% от числа всех пользователей. Изучив программирование на языке высокого уровня (HLL – high-level language), Вы должны уже достаточно хорошо представлять себе этот уровень архитектуры компьютера. Попробуйте, используя Ваш программистский опыт, сами сформулировать отличия ви́дения архитектуры ЭВМ на этом уровне, по сравнению с предыдущим уровнем пользователей-непрограммистов.[2]
3. Системные программисты (system programmers). Это самая небольшая (значительно меньше 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* (СПО – system software), которое в основном предназначено для автоматизации процесса программирования. Это уже упомянутые *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

[1] Отметим, что часто термин «архитектура», применяется для компьютера и в более узком смысле, например, «архитекутура системы команд» (ISA – Instruction Set Architecture).

[2] К этому уровню часто относят и пользователей, которые не используют в своей работе языки программирования, а создают прикладное программное обеспечение с помощью специальных инструментальных средств, например, конструкторов Web сайтов.

программирования, по аналогии с промышленным производством, можно образно сравнить со *средствами производства* остальных программ. В литературе по архитектуре ЭВМ этот внутренний уровень обычно разделяют на два подуровня: операционной системы (3а) и набора машинных команд (Ассемблера) (3б), который часто называют ISA (Instruction Set Architecture).

Разумеется, можно выделить и другие уровни ви́дения архитектуры компьютера, не связанные с его *использованием*. В качестве примеров можно указать *уровень микроархитектуры*, на котором рассматриваются правила выполнения компьютером машинных команд, и *логический уровень* инженера-разработчика аппаратуры компьютера. На следующем, *инженерном* (или *физическом*) уровне, рассматриваются радиотехнические схемы для реализации логических схем, исследуются новые материалы для построения схем ЭВМ и т.д. Эти уровни изучаются на других специальностях, и не будут нами подробно рассматриваться. В этой книге изучаются в основном второй и третий уровни, но иногда, в качестве примеров, совсем немного будет рассказано и о более глубоких уровнях ви́дения 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.

Часто эти описания очень громоздки. Например, для процессоров фирмы Intel это «Intel 64 and IA-32 Architecture Software Developer's Manual», пять больших книг от 400 до 1000 страниц каждая. Злые языки говорят, что число слов в документации пропроционально числу транзисторов в процессоре 🙂. Описание машинных команд при этом достаточно строгое (приведён так называемый микрокод, описывающий алгоритм выполнения сложных команд).

2. В качестве другого способа описания архитектуры компьютера на *внутреннем* уровне можно с 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 нам 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 Вам уже должно быть 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. [i] [см. сноску в конце главы]. В качестве небольшого примера, однако, в виде дополнительного материала 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 небольших аппаратных ресурсов такого учебного компьютера).

Вы, конечно, уже знаете, что сейчас производятся самые разные компьютеры. Определим теперь, архитектура *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.[1] О других способах организации ЭВМ, в частности, о так называемых аналоговых (непрерывных) и квантовых компьютерах, можно совсем немного (как говорится, «для общего развития») почитать в 15-й главе этой книги.

---

[1] В последнее время универсальные процессоры всё чаще работают вместе со *специализированными сопроцессорами*, которые более эффективно обрабатывают конкретные виды данных. Это, например, графические, тензорные, нейропроцессоры и др.

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. При описании ЭВМ машина фон Неймана выполняет примерно ту же роль, как и машина Тьюринга при описании алгоритмов и их исполнителей.
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 будет 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 конкретного 64-битного компьютера x86-x64 фирмы Intel, модели этой ЭВМ широко используются в настоящее время.
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

Но, увы, самые очевидные вещи как раз хуже всего и доходят до сознания людей.
Айзек Азимов. «Я, робот»

1. Для чего необходимо выделять различные уровни ви́дения архитектуры компьютера ?
2. Приведите примеры прикладных и системных программ, которые Вы используете на своем компьютере.
3. Сформулируйте различия в уровнях ви́дения архитектуры компьютера для конечного пользователя и прикладного программиста.
4. Чем отличаются концептуальный и внутренний уровни ви́дения архитектуры компьютера ?
5. Для чего предназначены системы программирования и операционные системы ?
6. Почему знание языка Ассемблера помогает в понимании архитектуры ЭВМ ?

---

[i] Для продвинутых читателей. В качестве примера можно назвать достаточно широко распространенные языки высокого уровня описания и моделирования электронных устройств Verilog, с синтаксисом, похожим на язык C, и язык VHDL (Very high speed integrated circuits Hardware Description Language), с синтаксисом, похожим на язык Ада. Например, вот описание некоторой логической схемы на языке Verilog:

```
module F(input logic a,b,c, output logic y);
    assign y = ~a & ~b & ~c |
    a & ~b & ~c |
    a & ~b & c;
/* Это y:=not a and not b and not c or
           a and not b and not c or
           a and not b and c */
endmodule
```