

Глава 2. Машина фон Неймана

Если люди отказываются верить в простоту математики, то это только потому, что они не понимают всю сложность жизни.

Джон фон Нейман

В конце 40-х и начале 50-х годов прошлого века во многих научных центрах велись разработки электронных вычислительных машин [3,24]. Можно отметить первую ламповую ЭВМ ENIAC, построенную в 1945 году сотрудниками Пенсильванского университета США Дж. Эккертом (Eckert John Presper, Jr.) и Дж. Моучли (John Mauchly), разработанную в 1951 году в Киеве под руководством академика С.А. Лебедева МЭСМ (Малую Электронную Счётную Машину) и другие проекты.



Джон Фон Нейман
(1903-1957)

Большой вклад в дело развития вычислительной техники внёс талантливый математик и физик венгерского происхождения Джон фон Нейман(н) (John von Neumann). Во время мировой войны он участвовал в знаменитом Манхэттенском проекте по созданию атомной бомбы и хорошо понимал важность машин для проведения больших объёмов математических вычислений.

После войны фон Нейман был профессором Принстонского университета США (заметим, что в это же время и тоже профессором там работал сам А. Эйнштейн). В 1945 году, ознакомившись с ЭВМ ENIAC, фон Нейман занялся проектированием нового компьютера с названием EDVAC (Electronic Discrete Variable Automatic Computer). Вскоре в соавторстве с Артуром Берксом (Arthur Burks) и Германом Голдстейном (Herman Goldstine) он описал в техническом докладе эту ЭВМ, обладающую рядом новых особенностей. Со временем стало ясно, что эти особенности желательно включать в архитектуру всех разрабатываемых в то время компьютеров. А вскоре пришло и понимание того, что эти новые свойства вычислительных машин, по сути, описывают архитектуру некоторого абстрактного универсального вычислителя, который сейчас принято называть *машиной фон Неймана*.¹

Эта машина является *абстрактной моделью* ЭВМ, однако она отличается от абстрактных исполнителей алгоритмов (например, от хорошо известной машины Тьюринга). Машина Тьюринга может обрабатывать входные слова любой длины, поэтому её принципиально нельзя реализовать. Машина фон Неймана не поддаётся реализации по другой причине: многие детали в архитектуре этого вычислителя *не конкретизированы*. Это сделано специально, чтобы не сковывать творческого подхода к делу у инженеров-разработчиков новых ЭВМ. Можно сказать, что машина фон Неймана рассматривается не на внутреннем, а только на концептуальном уровне видения архитектуры.

В некотором смысле машина фон Неймана подобна *абстрактным структурам данных*. У абстрактных структур данных, например, двоичных деревьев, для их использования необходимо предварительно выполнить конкретную реализацию: произвести отображение на структуры данных в некотором языке программирования, а также реализовать соответствующие операции над этими данными.

Можно сказать, что в машине фон Неймана зафиксированы те прогрессивные особенности архитектуры, которые в той или иной степени должны были быть присущи всем компьютерам того времени. Разумеется, практически все современные ЭВМ по своей архитектуре в значительной степени отличаются от машины фон Неймана. Эти отличия, однако, удобно изучать именно как *отличия*,

¹ Документ, описывающий проект EDVAC, первоначально не предназначался для публикации. Вскоре, однако, ксерокопии этой работы были разосланы Г. Голдстейном (он был куратором проекта от армии США) во многие университеты и научные центры, и на первой странице почему-то стояла только фамилия фон Неймана [John von Neumann. First Draft of a Report on the EDVAC. 30 June 1945]. Недоброжелатели потом говорили, что фон Нейман был всего лишь научным редактором упомянутой выше работы. Полностью (под тремя фамилиями) работа опубликована в 1946 году [2]. Своё авторство на схему "машины фон Неймана" предъявили упоминавшиеся ранее создатели ЭВМ UNIVAC Дж. Эккерт и Дж. Моучли. Позже это даже привело к судебному разбирательству, подтвердившему, что они тоже имеют права авторства на эту схему. Надо однако заметить, что в науке фон Нейман был значительно более крупной величиной, как никак член Национальной АН и Академии Искусств и Наук США 😊.

проводя сравнения и сопоставления с машиной фон Неймана. Мы тоже часто будем обращаться внимание на отличия машины фон Неймана от современных ЭВМ. основополагающие свойства архитектуры машины фон Неймана сформулированы в виде **принципов фон Неймана**. Эти принципы в значительной степени определяли основные черты архитектуры двух первых *поколений* ЭВМ [3].¹

На рис. 2.1 приведена схема машины фон Неймана, как она изображается в учебниках, посвященных архитектуре ЭВМ.² На этом рисунке толстыми (двойными) стрелками показаны *потоки команд и данных*, а тонкими – передача между отдельными устройствами компьютера *управляющих и информационных сигналов*.³ Как вскоре станет ясно, выполнение каждой команды приводит к выработке последовательности управляющих сигналов, которые заставляют узлы компьютера совершать те или иные действия. С помощью же информационных сигналов одни узлы компьютера сообщают другим узлам о том, что они успешно выполнили действия, предписанные управляющими сигналами, либо зафиксировали ошибки в своей работе.

Как видно из приведённого рисунка, машина фон Неймана состоит из памяти (мемогу – этот термин впервые введён фон Нейманом!), устройств ввода/вывода и *центрального процессора*

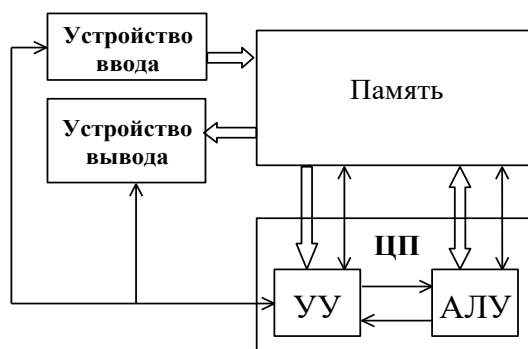


Рис. 2.1. Схема машины фон Неймана.

(ЦП). Заметим, что в начальный период развития вычислительной техники он назывался просто процессором. Позже, когда в составе ЭВМ, кроме основного, появились и другие процессоры, этот основной процессор и стали называть центральным процессором (CPU – Central Processing Unit). Для многоядерных ЭВМ сейчас обычно вместо процессоров говорят о процессорных ядрах.

Центральный процессор, в свою очередь, состоит из *устройства управления (УУ)* и *арифметико-логического устройства (АЛУ)*. Заметим, что первые ЭВМ работали только с числами, поэтому АЛУ тогда называли просто арифметическим устройством.

Сейчас будут последовательно рассмотрены все узлы машины фон Неймана и выполняемые ими функции, при этом будут формулироваться принципы фон Неймана (они выделяются **жирным шрифтом**). Основные понятия при этом выделяются *курсивом*.

¹ Как уже упоминалось, в послевоенные годы в СССР под руководством академика С.А. Лебедева проводились работы по созданию первых отечественных ЭВМ. При этом С.А. Лебедев, независимо от фон Неймана, также сформулировал аналогичные и даже более детальные принципы построения вычислительных машин, но из-за секретности таких работ в СССР они не были опубликованы в открытой печати.

² У нас рассматривается немного модифицированная схема машины фон Неймана. Здесь устройства ввода/вывода обмениваются данными непосредственно с памятью, а в схеме самого фон Неймана этот обмен производился через арифметико-логическое устройство (АЛУ) под контролем устройства управления. Например, при чтении данных устройство ввода передавало их в регистр АЛУ, которое уже затем записывало эти данные в память. Таким образом, при выполнении ввода/вывода центральный процессор не мог выполнять никакие другие команды, то есть предполагается блокировка работы ЦП во время выполнения операций ввода/вывода. Кроме того, в машине фон Неймана предполагалось и наличие внешней памяти. Забегая вперед, можно сказать, что рассматриваемая здесь модифицированная машина фон Неймана близка к современным реализациям, когда внешние устройства подключаются напрямую к памяти, минуя центральный процессор.

³ В современной терминологии это внутренние шины процессора, соединяющие его основные узлы. В этих шинах различают линии (провода) для передачи команд и обрабатываемых данных, а также адресные и управляющие линии.

2.1. Память

Память определяет быстроедействие.

Джон фон Нейман

Записывая что-то в память ЭВМ, помните, куда Вы это положили.

Первая аксиома Лео Бейзера

Принцип линейности и однородности памяти. *Память* машины фон Неймана – это линейная (упорядоченная) и однородная последовательность некоторых элементов, называемых *ячейками* (memory cells). В любую ячейку памяти другие устройства машины (по толстым стрелкам на рис. 2.1) могут записывать и считывать информацию, причем время чтения из любой ячейки одинаково для всех ячеек памяти. Время записи в любую ячейку тоже одинаково (это и есть принцип *однородности* памяти). Время чтения из ячейки памяти, однако, может не совпадать со временем записи в неё. Такая память в современных компьютерах называется *памятью с произвольным доступом* RAM (Random Access Memory).

Современные ЭВМ, однако, могут иметь участки памяти разных видов. Например, некоторые области памяти поддерживают только чтение (по-английски эта память называется ROM (Read Only Memory), данные в такую память записываются один раз при изготовлении этой памяти, они сохраняются и при отключении электрического напряжения. Другие виды памяти могут допускать запись, но за большее время, чем в остальную память (это так называемая *полупостоянная* память, на которой обычно реализованы схемы для хранения служебных программ, например, BIOS), флэш-память и др.

Ячейки памяти в машине фон Неймана нумеруются от нуля до некоторого положительного числа N (это и означает, что память *линейная*), причём число N в "настоящих" ЭВМ часто пропорциональна степени двойки, минус единица. *Адресом* ячейки называется её номер. Каждая ячейка состоит из более мелких частей, именуемых *разрядами*, они тоже нумеруются от нуля и до определенного числа. Количество разрядов в ячейке обозначает *разрядность памяти*. Каждый разряд может хранить одну *цифру* в некоторой системе счисления. В большинстве ЭВМ используется двоичная система счисления, т.к. это более выгодно с точки зрения инженерной реализации. В этом случае каждый разряд хранит одну двоичную цифру или один *бит* информации. Восемь последовательных бит составляют один *байт*.¹ Сам фон Нейман тоже был сторонником использования двоичной системы счисления, что позволяло хорошо описывать архитектуру узлов ЭВМ с помощью логических (булевских) выражений.

Содержимое ячейки называется *машинным словом*. С точки зрения архитектуры, машинное слово – это минимальный объём данных, которым могут обмениваться между собой различные узлы машины по толстым стрелкам на рис. 2.1 (не надо, однако, забывать о передаче сигналов по тонким стрелкам). Из каждой ячейки памяти можно считать *копию* машинного слова и передать её в другое устройство компьютера, при этом оригинал не меняется. При записи в ячейку её старое содержимое пропадает и заменяется новым машинным словом.

Заметим, что на практике решение задачи сохранения исходного машинного слова при чтении из ячейки для некоторых видов памяти является нетривиальным и достаточно трудоёмким. Дело в том, что в такой памяти, которая называется *динамической* памятью с произвольным доступом DRAM (Dynamic RAM), при чтении оригинал разрушается, и его приходится каждый раз восстанавливать после чтения данных. Использование такой памяти экономически более выгодно, она дешевле. Биты в этой памяти хранятся в крошечных конденсаторах, поэтому хранимые в динамической памяти данные с течением времени разрушаются и сами по себе. Микроконденсаторы ячеек быстро теряют электрический заряд, поэтому приходится часто (примерно каждые 30 миллисекунд) восстанавливать содержимое этой памяти. Ничего страшного в этом нет, сейчас при работе микросхемы динамической памяти тратят на эту регенерацию примерно 1% своего времени.

¹ Говорят, что пираты Вест-индских островов для совершения мелких платежей разрубали золотой пиастр или дублон на 8 равных частей, которые и назывались *битами*. В 1948 г. математик Джон Уайлдер Таки (John Wilder Tukey) вместо словосочетания *binary digit* (двоичная цифра), стал применять короткое *bit*. Впервые байт из 8 бит появился в семействе ЭВМ IBM System/360 в 1964 году.

В компьютерах может использоваться и другой вид памяти, которая называется *статической* памятью SRAM (Static RAM), при чтении из неё и при хранении данные не разрушаются (пока подаётся электрическое питание). Статическая память работает быстрее, чем динамическая, однако она в несколько раз дороже, т.к. требует для своей реализации больше электронных схем (2-3 транзистора на бит для динамической памяти и 6-8 транзисторов на бит для статической).¹ [см. сноску в конце главы].

Память современных ЭВМ работает весьма надёжно, ошибка обмена данными для одной микросхемы такой памяти случается примерно раз в 10 лет (одна ошибка в полгода на средний компьютер). Однако на компьютерах со значительным объёмом оперативной памяти, например, файловых серверах, ошибки могут случаться раз в месяц, а для современных ПК примерно раз в несколько месяцев. Если такой уровень ошибок неприемлем (например, в компьютерах для военного применения), то оперативная память снабжается схемами обнаружения двойных и коррекции одиночных ошибок ЕСС (Error Correction Code). Эти схемы в несколько раз повышают надёжность памяти, но увеличивают её стоимость примерно на 10%. В то же время стоит отметить, что оперативная память менее надёжна, чем процессор, так как содержит значительно больше электронных схем.¹

Приведём типичные *характеристики памяти* современных ЭВМ.

1. Объём памяти – от нескольких до десятков миллиардов ячеек (обычно восьмиразрядных).
2. Скорость работы памяти: это *время доступа* (access time – минимальная задержка на чтение слова из памяти на некоторый регистр) и *время цикла* (cycle time – минимальная задержка на повторное чтение из *той же самой* ячейки памяти) – несколько наносекунд (1 секунда = 10^9 нс.). Следует отметить, что для упомянутой выше динамической памяти время цикла может быть *больше*, чем время доступа, так как надо ещё восстановить разрушенное при чтении содержимое ячейки.

Чтобы почувствовать, насколько мало это время, надо учесть, что за одну наносекунду электромагнитный сигнал (или, как часто не совсем правильно говорят, *свет*) проходит в пустоте около 30 сантиметров (а в медных проводах всего 22 сантиметра). Следовательно, быстрые компьютеры просто не могут быть слишком большими, при таком времени доступа расстояние от процессора до оперативной памяти не должно превышать 10 сантиметров! Если уменьшить время доступа ещё в десять раз, то вся центральная часть мощного компьютера должна размещаться в кубике размером несколько сантиметров. Заглянув внутрь системного блока компьютера можно заметить, что продолговатые планки оперативной памяти стоят совсем близко от коробочки процессора (на которой обычно расположен вентилятор). Вероятно, конструкторы ЭВМ с тоской вспоминают время, когда одна машина занимала большой зал со шкафами, набитыми электроникой. Впрочем, современные супер-ЭВМ опять таки стали *очень* большими 😊.

3. Стоимость. Для основной памяти ЭВМ пока достаточно знать, что чем быстрее такая память, тем она, естественно, дороже. Конкретные значения стоимости памяти меняются весьма быстро с развитием вычислительной техники.

Принцип неразличимости команд и данных. С точки зрения программиста машинное слово представляет собой либо команду, либо подлежащее обработке данное (это число, символьная информация, элемент изображения и т.д.). Для краткости в дальнейшем будем называть такую информацию "числами". Данный принцип фон Неймана заключается в том, что числа и команды *неотличимы* друг от друга – в памяти и те и другое представляются некоторым набором разрядов, причем по внешнему виду машинного слова нельзя определить, что оно собой представляет – команду или число.

¹ Например, динамическая RAM объёмом 4 Гб содержит $2 * (4 * 8 * 2^{30}) = 2^{36}$ транзисторов, а современный процессор только примерно $10^9 \approx 2^{30}$ транзисторов. С другой стороны, оперативная память имеет значительно более регулярную структуру, чем процессор. Интересно, что, по мнению многих исследователей, подавляющее большинство ошибок оперативной памяти вызывается космическими лучами, поэтому глубоко под землей (или под специальной защитой, например такой, как у ядерных реакторов) память работает значительно надёжнее. Особенно это относится к статической памяти SRAM, у которой ёмкость заряда на бит значительно меньше, чем у DRAM. Как следствие, такая память должна иметь особые, так называемые многобитные (например, с названием Chipkill), схемы по защите от ошибок, которые уменьшают частоту неисправленных ошибок примерно в 10 раз.

Из неразличимости команд и данных как следствие вытекает принцип хранимой программы. Этот принцип является очень важным, его суть состоит в том, что программа хранится в памяти вместе с числами.¹ Чтобы понять важность этого принципа рассмотрим, а как программы вообще появляются в памяти машины. Понятно, что, во-первых, команды программы могут, наравне с числами, вводиться в память "из внешнего мира" с помощью устройства ввода (этот способ был основным в первых компьютерах). А теперь надо вспомнить, что на вход алгоритма можно в качестве входных данных подавать запись некоторого другого алгоритма (в частности, свою собственную запись). Остается сделать последний шаг в этих рассуждениях и понять, что *выходными* данными алгоритма тоже может быть запись некоторого другого алгоритма.² Таким образом, одна программа может в качестве результата своей работы поместить в память компьютера другую программу. Как Вы уже вероятно знаете, именно так и работают *компиляторы*, переводящие (транслирующие) программы с одного языка на другой.

Следствием принципа хранимой программы является то, что программа, может *изменяться* во время счёта самой этой программы. В частности, такая программа может *самомодифицироваться*, то есть сама изменять себя во время счёта. В настоящее время саомодифицирующиеся программы применяются редко, например, это любят делать так называемые полиморфные (самоизменяющиеся) вирусы. В то же время на первых ЭВМ, как Вы увидите далее, использование саомодифицирующихся программ часто было единственным способом реализации некоторых видов алгоритмов.

Заметим также, что, когда фон Нейман (с соавторами) писал техническое задание на свою машину, многие из тогдашних ЭВМ хранили программу в памяти одного вида, а числа – в памяти другого вида, поэтому этот принцип являлся в то время революционным. В современных ЭВМ и программы, и данные, как правило, хранятся в одной и той же памяти.³

2.2. Устройство Управления

Проще всего управлять обществом, члены которого отвергают саму идею объединения.

Стивен Кинг. «Тёмная башня»

Власть – палка о двух концах.

Фрэнк Хербер. «Дюна»

¹ Этот принцип придумал не фон Нейман. Первая машина с такой памятью называлась Манчестерская малая экспериментальная машина SSEM (Small-Scale Experimental Manchester Machine), созданная в Манчестерском университете ещё в 1948 году. Память для этой машины разработал Фредерик Уильямс (Williams Frederic Calland), это были так называемые ртутные линии задержки – длинные металлические трубки, наполненные парами ртути, в этих парах распространялись звуковые волны кодирующие нули и единицы. По длине трубки укладывалось несколько сотен бит, скорость доступа была примерно 200 мкс. Заметим, что эта память не совсем отвечала принципу однородности памяти фон Неймана, так как одинаковое время доступа к каждой ячейке было только *в среднем* за большое число обращений. В дальнейшем он же разработал более перспективную память на электронно-лучевых трубках.

Такие ЭВМ, как принято говорить, имеют *Пристонскую* архитектуру (так как фон Нейман тогда работал в Пристонском университете), в отличие от альтернативной *Гарвардской* архитектуры, в которой была отдельная память для команд и для чисел. Сейчас это устаревшие термины, так как все ЭВМ имеют общую память для команд и чисел (кроме так называемой кэш-памяти первого уровня, которая часто делается отдельной для команд и данных, об этой памяти будет рассказано в другой главе).

² Любопытно, что можно написать программу, которая *ничего не вводит* и выводит запись своего собственного текста. Вы можете попробовать сделать это на некотором языке программирования высокого уровня (например, на Паскале). Кому любопытно, могут посмотреть, как это делается, в книге [18]. Такие программы называются куайнами (quine) по имени философа, логика и математика Уилларда Ван Ормана Куайна (Willard Van Orman Quine). Первая такая программа, обладающая возможностью само репродуцирования, была написана в 1960 году Хэмишем Дюаром (Hamish Dewar), на языке Atlas Autocode и имела длину в 26 строк. Возможны и более сложные варианты таких программ (куайн n-го порядка, цепной куайн и т.д.).

³ В современных ЭВМ некоторые программы, называемые Базовыми процедурами ввода/вывода (BIOS) крайне нежелательно изменять (стирать) во время работы компьютера, а также и после выключения питания. Такие программы располагают в уже упомянутой ранее памяти типа ROM, закрытой для записи и сохраняющей данные при отсутствии электрического питания, однако и в этой памяти команды тоже не отличимы от чисел.

Как ясно из названия, устройство управления (УУ) *управляет* всеми остальными устройствами ЭВМ. Оно осуществляет это путем посылки *управляющих сигналов*, подчиняясь которым остальные устройства производят определенные действия, предписанные этими сигналами. Следует обратить внимание, что это устройство является единственным, от которого на рис. 2.1 отходят тонкие стрелки ко *всем* другим устройствам. Остальные устройства на этой схеме могут "командовать" только памятью, делая ей запросы на чтение и запись машинных слов.

Принцип автоматической работы. Этот принцип называют ещё принципом **программного управления**. Машина, выполняя записанную в её памяти *программу*, функционирует автоматически, без участия человека, если только такое участие не предусмотрено в самой программе (например, при вводе данных с клавиатуры). Пример устройства, которое может выполнять команды, как и ЭВМ, но не в автоматическом режиме – обычный (непрограммируемый) калькулятор. Последовательность команд для калькулятора непосредственно задаёт сам человек.

Программа – множество записанных в памяти (не обязательно последовательно) машинных команд, задающих действия, описывающих шаги работы алгоритма. Команды программы обрабатываются хранимые в памяти компьютера данные. Таким образом, программа – это запись алгоритма на *языке машины*. *Язык машины* – набор всех возможных операций, выполняемых командами и допустимые форматы этих команд. Каждая команда однозначно определяется своим *кодом операции* (в простейшем случае это просто номер команды). Например, язык одной из наших учебных машин УМ-3 будет содержать всего 25 кодов операций, в машинном языке младшей модели 8086 фирмы Intel 135 команд, а число команд современных компьютеров этой фирмы приближается к тысяче.¹

Принцип последовательного выполнения (последовательной работы). Устройство управления выполняет некоторую команду *от начала до конца*, а затем по определенному правилу выбирает следующую команду для выполнения, затем следующую и т.д. При этом каждая команда либо сама явно указывает на команду, которая будет выполняться за ней, (такие команды называются командами перехода), либо следующей будет выполняться команда из следующей ячейки памяти. Этот процесс продолжается, пока не будет выполнена специальная команда останова, либо при выполнении очередной команды не возникнет *аварийная ситуация* (например, деление на ноль). Аварийная ситуация – это аналог *безрезультативного* останова алгоритма, например, для машины Тьюринга это чтения из клетки ленты символа, которого нет в заголовке ни одной колонки таблицы.

2.3. Арифметико-логическое Устройство

Лучший способ в чём-то разобратся до конца – это попробовать научить этому компьютер.

Дональд Эрвин Кнут

В машине фон Неймана арифметико-логическое устройство (АЛУ) может выполнить следующие действия.

1. Читать содержимое ячейки памяти (машинное слово), т.е. поместить копию этого машинного слова в заданную ячейку, расположенную в самом АЛУ. Такие ячейки памяти, расположенные не в основной памяти, а в других устройствах ЭВМ, называются *регистровой памятью* или просто *регистрами*. Таким образом, АЛУ может читать машинное слово из памяти на один из своих регистров.
2. Записать копию содержимого одного из своих регистров в некоторую ячейку памяти. Когда не имеет значения, какая операция (чтение или запись) производится, говорят, что происходит *обмен* машинным словом между регистром и основной памятью ЭВМ. Таким образом, как уже говорилось, машинное слово – это минимальная порция информации для обмена между регистрами и основной памятью.
3. АЛУ может также выполнять различные *операции* над данными в своих регистрах, например, сложить содержимое двух регистров, обычно называемых регистрами *первого* R1 и *второго* R2 операндов, и поместить результат этой операции на третий регистр, называемый в русскоязычной литературе, как правило, *сумматором* S. В англоязычной литературе этот регистр называ-

¹ Не надо пугаться, большая часть этих команд предназначена для весьма специфических векторных и мультимедийных операций.

ется Accumulator и сокращается до буквы А, с этим обозначением Вы столкнетесь далее при изучении языка Ассемблера.

2.4. Взаимодействие УУ и АЛУ

Компьютер делает не то, что Вы хотите, чтобы он сделал, а то, что Вы ему приказываете.

Третий закон Грира

Революционность идей фон Неймана заключается в строгой *специализации*: каждое устройство компьютера отвечает за выполнение только своих функций. Например, раньше память ЭВМ часто не только хранила данные, но и могла производить операции над ними.¹ Теперь же было предложено, чтобы память только хранила данные, АЛУ производило арифметико-логические операции над данными в своих регистрах, устройство ввода вводило данные из "внешнего мира" в память и т.д. Таким образом, фон Нейман предложил жёстко распределить выполняемые ЭВМ функции между различными устройствами, что существенно упростило схему машины, и сделало более понятным её работу.

Устройство управления тоже имеет свои регистры, оно может считывать команды из памяти на специальный *регистр команд* RK (в англоязычной литературе IR – instruction register), на котором всегда хранится *текущая* выполняемая команда. Регистр УУ с именем RA называется *регистром* или *счётчиком адреса* (в англоязычной литературе его часто обозначают IP – instruction pointer). *Перед* выполнением текущей команды УУ записывает в него адрес *следующей* команды² (первая буква в сокращении слова регистр в дальнейшем изложении обозначается латинской буквой R).

Рассмотрим, например, схему выполнения команды, реализующей оператор присваивания для сложения двух чисел $z := x + y$. Здесь x , y и z – адреса (номера) ячеек памяти, в которых хранятся, соответственно, операнды и будет помещен результат операции сложения (предположим, что такая команда есть в языке машины). После получения из памяти этой команды на регистр команд RK, УУ последовательно посылает управляющие сигналы в АЛУ, предписывая ему сначала считать операнды x и y из памяти и поместить их на регистры R1 и R2. Затем по следующему управляющему сигналу АЛУ производит операцию сложения чисел, находящихся на регистрах R1 и R2, и записывает результат на регистр сумматора S. По следующему управляющему сигналу АЛУ пересылает копию регистра S в ячейку памяти с адресом z .³ Ниже приведена иллюстрация описанного примера с помощью операторов присваивания, где R1, R2 и S – переменные, обозначающие регистры АЛУ, ПАМ – массив ячеек, обозначающий оперативную память ЭВМ.

$R1 := \text{ПАМ}[x]; R2 := \text{ПАМ}[y]; S := R1 + R2; \text{ПАМ}[z] := S$

В дальнейшем конструкция $\text{ПАМ}[A]$, как это принято в компьютерной литературе, будет обозначаться как $\langle A \rangle$, тогда схема выполнения команды переписывается так:

$R1 := \langle x \rangle; R2 := \langle y \rangle; S := R1 + R2; \langle z \rangle := S$

Таким образом, процессор умеет складывать числа не в оперативной памяти, а только на регистрах. Рассмотренный подробный (пошаговый) способ выполнения ЭВМ машинных команд принято относить к так называемому уровню *микроархитектуры* компьютера [21]. Этот уровень является промежуточным между уровнем машинных команд и уровнем электронных схем.

Опишем теперь более формально шаги выполнения (такт работы) одной команды в машине фон Неймана:

1. $RK := \langle RA \rangle$; считать из ячейки памяти с адресом из регистра RA команду на регистр команд RK.

¹ Любопытно отметить, что всё развивается по спирали и сейчас разработаны формальные вычислители с активной памятью, которая не только хранит, но и обрабатывает данные. В качестве примера можно упомянуть описанную в 2014 году Фабио Траверсой (Fabio L. Traversa) Universal Memcomputing Machine (UMM), в ней нет АЛУ, а вся обработка данных производится прямо в памяти. Впрочем, UMM эквивалентна машине Тьюринга.

² Этот следующий адрес может далее быть изменён самой выполняемой командой, такие команды, как уже говорилось, называются командами переходов.

³ В схеме на рис. 2.1 от АЛУ к УУ тоже ведёт тонкая стрелка, однако она определяет не управляющий сигнал (так как АЛУ не может "командовать" УУ), а информационный, с помощью таких сигналов АЛУ "рапортует" УУ, что заданное действие выполнено, или при его выполнении возникла ошибка.

2. $RA := RA + 1$; увеличить счетчик адреса на единицу.
3. Выполнить очередную команду, хранящуюся в регистре RK .

Затем по такой же схеме из трёх шагов выполняется следующая команда и т.д. Заметим, что после выполнения очередной команды ЭВМ полностью "забывает", какую именно команду она только что выполнила. Отметим, что по такому же принципу выполняли свои "команды" (шаги алгоритма) и такие известные абстрактные исполнители алгоритмов, как машина Тьюринга и Нормальные алгоритмы Маркова.

Итак, если машинное слово попадает на регистр команд, то оно *интерпретируется* УУ как команда, а если слово попадает в регистр АЛУ, то оно *по определению* считается числом. Это позволяет, например, складывать команды программы как числа, либо выполнить некоторое число как команду. Разумеется, обычно такая ситуация является семантической ошибкой, если только специально не предусмотрена программистом для каких-то особых целей. Необходимость и способ обработки команд как чисел будет показан далее в одной из учебных машин.

Современные ЭВМ в той или иной степени нарушают практически все принципы фон Неймана. Не нарушаются только принцип автоматической работы (он лежит в самой основе определения ЭВМ как устройства для *автоматической* обработки данных), и принцип хранимой программы.

Например, существуют компьютеры, которые различают команды и данные. В них каждая ячейка основной памяти кроме собственно машинного слова хранит ещё специальный признак, называемый *тэгом* (tag), который и определяет, чем является это машинное слово.¹

Для экономии памяти современные компьютеры могут приписывать такой тэг не каждой ячейке в отдельности, а сразу целой последовательности ячеек, называемой сегментом, секцией или страницей. Таким образом, различают, например, секции команд и данных. Так нарушается принцип неразличимости команд и чисел. В такой архитектуре при попытке выполнить число как команду, либо складывать команды как числа, процессором может фиксироваться аварийная ситуация. Очевидно, что это усложняет архитектуру ЭВМ, но позволяет повысить надежность программирования на языке машины, не допуская, например, случайного "выхода программы на константы".

Практически все современные ЭВМ нарушают принцип однородности и линейности памяти. Память может, например, состоять из двух частей со своей независимой нумерацией ячеек в каждой такой части (с такой архитектурой Вы вскоре познакомитесь); или быть двумерной, когда адрес ячейки задается не одним, а двумя числами; либо ячейки памяти могут вообще не иметь адресов, такая память называется *ассоциативной*.²

Почти все современные компьютеры нарушают и принцип последовательного выполнения команд: они могут одновременно выполнять несколько команд как из одной, так и из разных программ. Такие компьютеры имеют несколько центральных процессоров, это в частности, так называемые многоядерные компьютеры, а также быть так называемыми *конвейерными* ЭВМ, которые будут рассмотрены далее в этой книге.

Среди принципов можно отметить и использование двоичной системы счисления для представления чисел.



Вообще говоря, следует отметить, что в архитектуре машины фон Неймана зафиксированы и другие принципы, которые из работы самого фон Неймана явно не вытекали, так как считались на то время *самоочевидными*. Так, например, предполагается, что во время выполнения программы не меняется число узлов компьютера и взаимосвязи между ними, не меняется число ячеек в оперативной памяти. Далее, считалось, что машинный язык при выполнении программы не

¹ Тэги длиной 3 бита в 48-битных ячейках успешно использовались в ЭВМ B6500 фирмы Burroughs 1966 года. Из современных можно отметить отечественную ЭВМ Эльбрус [31], система тэгов в этой машине близка к понятию типов данных в языках программирования высокого уровня. Например, была только одна (обобщённая) команда сложения, а тег операндов определял тип слагаемых (целые или вещественные, с одинарной или двойной точностью). Собственно, архитектура этой ЭВМ и поддерживала язык программирования высокого уровня Эль-76. В процессорах Intel вещественные регистры сопроцессора имеют теги из двух бит, описывающие содержание регистра (допустимое число, ноль, "не число", "пустой" регистр).

² Поиск нужной ячейки в ассоциативной памяти (content addressed memory) производится не по её адресу, а по содержимому хранящегося в этой ячейке машинного слова (например, считать из памяти первое машинное слово, хранящее целое число, кратное пяти; или слово, начинающееся с ключа 1234 и т.п.).

изменяется (например, "вдруг" не появляются новые машинные команды) и т.д.¹ В то же время сейчас существуют ЭВМ, которые нарушают и этот принцип. Во время работы одни устройства могут, как говорят, отбраковываться (например, отключаться для ремонта), другие – автоматически подключаться. Кроме того, во время работы программы могут, как изменяться, так и появляются новые связи между элементами ЭВМ (например, в так называемых *транспьютерах*). Существуют и компьютеры, которые могут менять набор своих команд, (правда, не во время, а только перед началом счёта программы) они называются ЭВМ с микропрограммным управлением, о чем немного рассказывается в предпоследней главе этой книги.

В заключение рассмотрения машины фон Неймана ещё раз обратим внимание на одно важное свойство компьютеров, которое часто ускользает от внимания читателей. Закончив выполнение текущей команды, машина начисто "забывает" о том, что это была за команда, где она располагалась в памяти, с какими операндами работала и т.д. И, хотя некоторые команды могут изменять значения специальных флагов (или регистра результата выполнения команды), но это не меняет сути дела, т.к. не сохраняется информации, какая именно команда, когда и как изменила эти флаги. Выполнение каждой следующей команды начинается "с чистого листа". Это важное свойство позволяет, например, надолго прерывать выполнение программы, запомнив относительно небольшой объём информации (текущее состояние регистров компьютера, сведения об открытых файлах и т.д.). В дальнейшем в любой момент возможно возобновление счета этой программы с прерванного места (разумеется, сама программа и её данные в памяти компьютера должны сохраниться, или же должны быть восстановлены в прежнем виде). Как будет показано далее, это свойство является основой для так называемого мультипрограммного режима работы компьютера.

На этом закончим краткое описание машины фон Неймана и принципов её работы. Первая ЭВМ, построенная по принципам фон Неймана, называлась EDSAC (Electronic Delay Storage Automatic Calculator – автоматический вычислитель с электронной памятью на линиях задержки) [3]. Компьютер EDSAC был построен в 1949 году в Кембриджском университете Морисом Уилксом (Moris Wilkes) при участии знакомого Вам по его знаменитой машине А. Тьюринга. EDSAC была одноадресной ЭВМ (что это такое Вы вскоре узнаете), которая работала в двоичной системе счисления со скоростью примерно 100 операций в секунду. Заметим, что именно от этой машины принято отсчитывать **первое** поколение ЭВМ (все предшествующие "не совсем настоящие" компьютеры можно условно отнести к нулевому поколению).

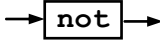
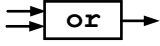
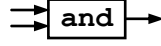
В конце этого раздела рассмотрим совсем немного архитектуру ЭВМ на уровне инженера-конструктора. Это будет сделано исключительно для того, чтобы снять тот покров таинственности с работы процессора, который есть сейчас у некоторых читателей: как же машина может автоматически выполнять различные операции над данными, неужели она такая умная? Такое чувство *непонимания* архитектуры на более низком уровне является для профессиональных программистов совершенно неприемлемым.

Аппаратура современных ЭВМ, если не учитывать оперативную память, конструируется из некоторых относительно простых элементов, чаще всего называемых в русскоязычной литературе (логическими) *вентильями* (по-английски – *logic gates*). Каждый вентиль является достаточно простой (электронной) схемой и реализует одну из логических операций, у него есть один или два *входа* (аргументы операции) и обычно один *выход* (результат). На входах и выходе могут быть электрические сигналы двух видов: низкое напряжение (трактуются как ноль или логическое значение **false**) и высокое (ему соответствует единица или логическое значение **true**).² Обычно для современных микросхем с напряжением питания около 1 вольта напряжение менее 0.4 вольта трактуется как логический

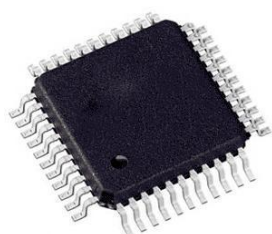
¹ В теории алгоритмов аналогом компьютера, нарушающего эти принципы, является, например, такая экзотическая модификация машины Тьюринга, у которой во время работы могут появляться и исчезать строки и столбцы её таблицы, а также изменяться содержимое клеток этой таблицы (движение влево заменять на движение направо и т.д.). Такие "запутанные" языки, на которых человеку очень трудно (а часто и невозможно) программировать, принято называть *эзотерическими* языками. В качестве примера можно привести язык Malbolge (так назывался один из кругов Ада в книге Данте Алигери "Божественная комедия"). Людям-программистам пока не удалось написать на нём ни одной программы, все известные программы на языке Malbolge получены другими программами (написанными на традиционных языках).

² В принципе вентили могут быть не только электрическими, но и, например, световыми, тогда на их вход по световодам подаются импульсы света различной интенсивности или поляризации. Такие устройства называются оптоэлектронными.

ноль, а напряжение более 0.8 вольта – как логическая единица. Можно выбрать такие основные вентили.

1. *Отрицание*, этот вентиль имеет один вход и один выход, он реализует хорошо известную Вам операцию отрицания **not** (НЕ) языка Паскаль. Другими словами, если на вход такого вентиля подается импульс высокого напряжения (значение **true**), то на выходе получится низкое напряжение (значение **false**) и наоборот. Далее в схемах этот вентиль изображается так:

2. *Дизъюнкция* или логическое сложение, этот вентиль реализует хорошо известную Вам операцию Паскаля **or** (ИЛИ), он будет изображаться как

3. И, наконец, вентиль, реализующий *конъюнкцию* или логическое умножение, в Паскале это операция **and** (И), будет изображаться как


Любую логическую функцию можно преобразовать к эквивалентному виду, в котором используются только три перечисленные выше логические операции. Здесь стоит заметить, что с чисто технической точки зрения легче создавать вентили, задающие логические операции **not and** (**nand** – штрих Шеффера) и **not or** (**nor** – стрелка Пирса), в их электронных схемах требуется на один транзистор меньше, чем для вентилях **and** и **or**. С помощью и **nand** и **nor** можно выразить все логические операции, поэтому для всех схем можно выбрать только один вентиль, например, **nand**.



Интегральная микросхема

Из вентилях строятся так называемые *интегральные схемы* (*integrated circuits, chips*)¹ – это набор вентилях, соединенных проводами и такими радиотехническими элементами, как сопротивления, конденсаторы и индуктивности, знакомые Вам из курса физики. В современных интегральных схемах, однако, роль сопротивлений, конденсаторов и индуктивностей часто тоже выполняют транзисторы, так что схема становится однородной, т.е. состоит только из транзисторов и проводников. Каждая интегральная схема реализует какую-нибудь функцию узла компьютера. Сейчас в научной литературе интегральные схемы, которые содержат порядка 1000 вентилях, называются малыми интегральными схемами (МИС), порядка 10000 вентилях – средними (СИС), порядка 100000 – большими (БИС), а число вентилях в сверхбольших интегральных схемах (СБИС) исчисляется миллионами и миллиардами.

Интегральная схема может иметь от нескольких десятков до тысяч внешних контактов. Например, процессор Intel Core i7 имеет на нижней поверхности 1366 контактов, расположенных с шагом около 1 мм. Большинство современных интегральных схем собираются на одной небольшой (прямоугольной) пластинке полупроводника с размерами порядка 1-2 сантиметра. Такая пластинка СБИС похожа на рельефный план большого города, со своими кварталами, домами, широкими проспектами и более узкими улицами. Это многоэтажный город, он может содержать до 9 изолированных друг от друга этажей-слоев со своими схемами и проводниками.

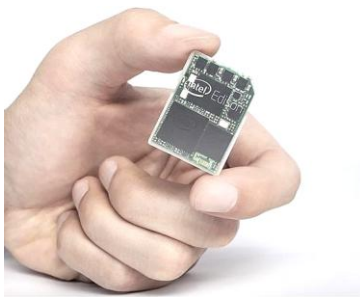


В объёме пары игровых костей уместится 6400 чипов Freescale Kinetis KL02

Впрочем, выпускаются и совсем миниатюрные ИС, например, фирма Freescale Semiconductors производит контроллерный чип Kinetis KL02 с размерами 1.9x2.0x0.4 мм. Несмотря на такие маленькие размеры, чип содержит 32-х разрядный процессор ARM Cortex-M0, 4 Кб оперативной и 32 Мб флэш-памяти. Из-за крайне низкой цены (около 70 центов на начало 2014 года) чип может встраиваться практически везде, от таблеток лекарств до бытовых вещей, например, обуви.

В несколько большем формате, размером с карту флэш-памяти, производятся уже "почти настоящие" ЭВМ. Например, в 2014 году фирма Intel начала производство компьютера Intel Edison для компактных и носимых устройств. Компьютер оснащён двухъядерным процессором Intel Atom с тактовой частотой 500 МГц, оперативной памятью 1 Гб, памятью пользователя 4 Гб, плюс разъем для microSD карт. Для связи с сетью имеются интерфейсы Wi-Fi и Bluetooth. Всё работает под управле-

¹ Первые интегральные схемы были созданы в 1958 году Джеком Килби (Jack Kilby) из компании Texas Instruments и одним из основателей фирмы Intel Робертом Нойсом (Robert Norton Noyce), на маленькой кремниевой пластинке размещалось несколько десятков транзисторов. Первая отечественная микросхема была создана в 1961 году в Таганрогском Радиотехническом Институте.

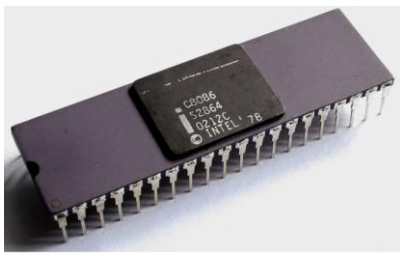


Компьютер Intel Edison, 2014 г.

нием ОС Linux. Имеются разъемы для подключения периферийных модулей. И все это размещается в размере карточки 35.5x25.0x3.9 мм. Такие компьютеры могут встраиваться практически во все носимые предметы (велосипедные шлемы, теннисные ракетки, кроссовки и т.д.).

Ещё меньшие размеры до 0.2x0.2x0.01 мм имеют микросхемы радиочастотной идентификации RFID (Radio Frequency IDentification), их можно встраивать внутрь бумажных банкнот, в швы одежды и т.д. Хотя для их работы и нужна достаточно большая антенна, но её можно, скажем, нанести в любом месте предмета бесцветной токопроводящей краской.

Обычно в качестве полупроводника в ИС используется очень чистый кремний, сейчас в нем допускается 1 атом посторонней примеси на 10^9 - 10^{10} атомов кремния (так называемая чистота "девять или десять девяток").¹



Процессор Intel 8086, 1978 г.

Дадим представление о количестве вентилях в различных электронных устройствах. Скажем, для того, чтобы реализовать схему самых простых электронных часов, необходимо порядка 1000 вентилях. Например, первый в мире 4-разрядный микроконтроллер Intel 4004 выпуска 1971 года содержал 2250 транзисторов. Из 10000 вентилях можно собрать простой процессор, например популярный Intel 8086, выпущенный в 1978 году, содержал 29000 транзисторов. Процессоры современных персональных ЭВМ реализуются на интегральной схеме, состоящей уже из сотен миллионов и миллиардов

транзисторов. Так, интегральная схема старого процессора Pentium 4 (выпуска 2002 года) содержала 42 миллиона транзисторов (это около 15 миллионов вентилях). Сейчас на одной пластинке кремния стали создавать не один, а несколько почти независимых друг от друга процессоров, собранные на базе таких интегральных схем компьютеры называются многоядерными.



Микропроцессор Oracle SPARC M8, 2017 год

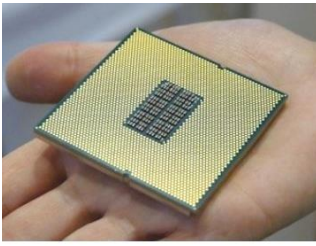
Примерно в 1989 году процессоры стали содержать более миллиона транзисторов на кристалле, а в 2004 году процессоры фирмы Intel имели уже около 500 млн. транзисторов. Первый процессор, содержащий более миллиарда транзисторов выпущен в 2006 году, а уже в 2008 году четырёхядерный процессор Tukwila, имел более 2 млрд. транзисторов. Таким образом, как и следует из знаменитого закона Гордона Мура (Gordon Moore), сформулированного в 1965 году, каждые 1.5-2 года количество транзисторов на кристалле удваивалось.² Примерно с 2015 года, однако, закон

Мура перестал действовать. В настоящее время число транзисторов на больших интегральных схемах составляет порядка 10 миллиардов. В качестве примера можно привести показанный слева 32-ядерный процессор SPARC M8 фирмы Oracle 2017 года выпуска, производимый по 20 нм. технологии, он "заточен" для работы с большими базами данных.

Большой проблемой в работе СБИС является тепловыделение (Thermal Power), например, уже поверхность ЦП Intel Pentium 4 при работе выделяла около 30 Вт/см². Для сравнения, сковородка, на которой жарится яичница, выделяет примерно 10 Вт/см² 😊. Впрочем, современные процессоры могут сохранять работоспособность до температуры примерно в 100 градусов. Выделяемая микропроцессором мощность называется его термопаке́том TDP (Thermal Design Power), обычно это от 15 до 90 и более ватт. От 20 до 50% этой мощности составляют паразитные токи утечки в микросхеме, они пропорциональны *четвёртой* (!) степени температуры микросхемы. Для контроля температуры по всей поверхности кристалла размещены температурные датчики (примерно по 10-15 на каждое процессорное ядро).

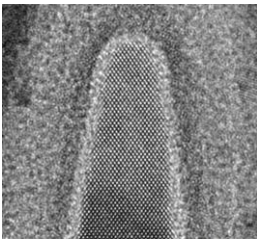
¹ Чтобы наглядно представить себе такую степень чистоты, вообразите, что Вы стоите на морском пляже, который простирается вдаль на целый километр и весь покрыт белым песком. Так вот, на поверхности всего этого пляжа Вы сможете найти только одну чёрную песчинку примеси.

² Вместе с законом Мура действует также так называемое правило Куми, его сформулировал в 2011 году профессор Стэнфорд Джонатан Куми (Jonathan Koomey). Это правило гласит, что объём вычислений на единицу затраченной энергии удваивается каждые 1.5-2 года.



Процессор Centriq 2400

Важной характеристикой интегральной схемы является так называемая проектная (технологическая) норма (technology node, feature size, critical dimension). Сейчас это примерно минимальная ширина проводника, соединяющего транзисторы (заметим, что общая протяженность проводников внутри микросхемы составляет порядка 10 километров, а плотность электрического тока в таком проводнике около 10^5 А/см²). В настоящее время для ЭВМ общего назначения в основном производятся процессоры с нормой 14 нанометров (нм.), но есть микросхемы с нормой 10 и даже 7 нм.ⁱⁱ [см. сноску в конце главы]. Например, слева показан процессор Centriq 2400 фирмы Qualcomm Datacenter Technologies (QDT) 2017 года, созданный по 10 нм. технологии. На кристалле 18 млрд. транзисторов, 48 64-битных ядер. На 2019 год некоторые фирмы анонсируют выпуск микросхемы с проектной нормой уже в 5 нм! Так как размер атома кремния составляет 0.24 нм., то это примерно 20 атомных слоёв.¹ Для сравнения, размер самого маленького вируса около 20 нм. (вирус гриппа 130 нм.), таким образом, на острие иглы можно разместить примерно 50 тыс. транзисторов.²



Затвор транзистора 22 нм
(точки - атомы кремния)

К сожалению, в обычный оптический микроскоп увидеть мелкие детали микросхемы не получится. Типичный размер отдельных элементов схемы (транзисторов и соединяющих их проводников) сейчас составляет около 14 нм., а длина волны фиолетового цвета (самого коротковолнового из различимых глазом человека) – около 380 нм. Теоретический предел разрешения составляет 1/4 длины волны, так что посмотреть на транзистор "глазами" не получится 😊. Образно говоря в оптический микроскоп можно различить не отдельные транзисторы-дома, а только целые большие "микрорайоны". Для получения изображения отдельного транзистора придётся использовать специальный *сканирующий* микроскоп, который особой, очень тонкой иглой "ощупывает" поверхность кристалла, что позволяет различать даже отдельные атомы. В качестве примера слева показано изображение верхней части канала транзистора (так называемого "плавника"), сделанного по 22 нм. технологии. Изображение получено при максимальном разрешении атомного сканирующего микроскопа. Точки в канале являются отдельными атомами кремния в его кристаллической решётке, по сторонам – полиморфный кремний. Ширина изолирующего слоя окиси кремния между ними около 1 нм., что составляет всего около 5 молекул SiO₂. Такая тонкая изоляция, даже учитывая, что SiO₂ почти идеальный диэлектрик, приводит к большим токам утечка, поэтому значительная часть потребляемой микросхемой мощности может бесполезно уходить в тепло.

Интересно, что у современных жёстких дисков (HDD) размер магнитной частицы, хранящей бит, тоже имеет размер около 10 нм., а головки чтения/записи "плавают" над поверхностью пластины на расстоянии около 15 нм.³ Скажем также, что современная 3D флэш-память также производится по 15 нм. технологии. Стоит также отметить, что и плотность записи на магнитную ленту в системах резервного копирования больших объёмов данных (стриммерах) составляет сейчас около 30 Гбит/кв.см. (200 Гбит/кв.дюйм). Ясно, что дальнейшее уменьшение этих размеров скоро натолкнётся на физический предел, за которым определяющими становятся уже законы квантовой физики.

В качестве примера сейчас будет рассмотрена очень простая интегральная схема сумматора, которая реализует функцию сложения двух одноразрядных двоичных чисел. Входными данными этой схемы являются значения одноразрядных переменных x и y , а результатом – их сумма, которая, в общем случае, является двухразрядным числом (обозначим разряды этого числа как a и b), формируемыми как результат сложения $x+y$. Если трактовать значения входных переменных x и y как

¹ Шаг в кристаллической решётке кремния 0.357 нм., однако кристаллическая решётка не кубическая, в так называемая гранцентрированная, так что на расстояние 5 нм. укладываются не 10, а 20 атомных слоёв.

² В 2019 году фирмы Cerebras Systems и TSMC выпустили по 16 нм. технологии микросхему размером 22x22 см., она содержит около 400000 процессорных ядер (небольшая часть из них отбраковывается за неисправностью) и 18 Гб встроенной SRAM. Общее число транзисторов примерно $1.2 \cdot 10^{12}$. Эта микросхема предназначена для работы в больших центрах обработки данных.

³ Поверхность самой пластины жесткого диска должна быть отполирована по 14-му (самому последнему!) классу чистоты. Чтобы оценить степень чистоты полировка пластины представьте, что мы увеличили размер диска так, чтобы длина дорожки составила 2000 км, тогда максимальный перепад высот на всём этом пути будет около 5 см. ⚠

логические значения (0 как **false** и 1 как **true**), то можно считать, что схема реализует некоторую логическую функцию с двумя аргументами и двухбитным результатом. Запишем таблицу истинности для этой функции:

x	y	b	a
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Легко вычислить, что выходные величины a и b будут определяться такими логическими формулами:

$$a = x \oplus y = x \text{ xor } y = (x \text{ or } y) \text{ and not } (x \text{ and } y)$$

$$b = x \text{ and } y$$

Реализуем схему двоичного сумматора¹ как набор вентилях, связанных проводниками (рис. 2.2. а). Здесь используется тот факт, что входной электрический сигнал (например, приходящий на вход x) с помощью проводов можно "продублировать" и одновременно подать на вход сразу двух вентилях. Заметим, что формально можно ввести отдельный вентиль "дублировать" с одним входом и двумя идентичными выходами, но не будем усложнять нашу схему.

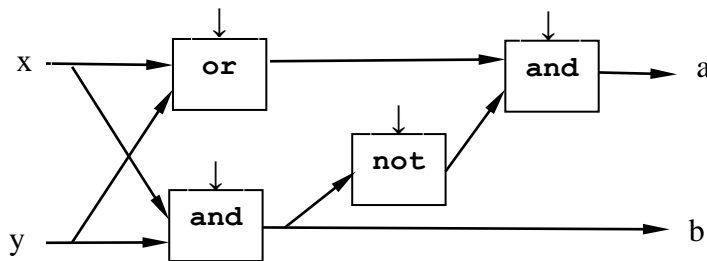


Рис. 2.2. а) Сборка двоичного сумматора из вентилях, ↓ – тактовые импульсы.

Каждый вентиль срабатывает (т.е. преобразует входные сигналы в выходные) не непрерывно, а только тогда, когда на этот вентиль по специальному управляющему проводу приходит так называемый *тактовый импульс* (clock cycle). Тактовые импульсы часто называются *синхронизирующими*, так как они задают моменты одновременного (синхронного) срабатывания вентилях схемы. Заметим, что по этому принципу работают так называемые *синхронные* логические схемы, в отличие от *асинхронных*, которые работают непрерывно (всё время). Суммирование чисел x и y в приведенной выше схеме осуществляется после последовательного прихода трёх тактовых импульсов (как говорят, за три такта).ⁱⁱⁱ [см. сноску в конце главы]

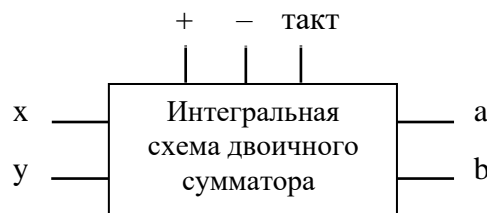


Рис. 2.2. б). Пример ИС двоичного сумматора.

Теперь, если реализовать этот двоичный сумматор в виде интегральной схемы (ИС), то она будет иметь не менее 7-ми внешних контактов (рис. 2.2 б). Это входные контакты x и y , выходные a и b , один контакт для подачи тактовых импульсов (о них уже говорилось, когда объяснялась работа вентилях), два контакта для подачи электрического питания (ясно, что без энергии ничего работать не будет) и, возможно, другие контакты.

Заметим, что основная память современных компьютеров также реализуется в виде набора нескольких сверхбольших интегральных схем. Пользователи, заглядывавшие внутрь персонального

¹ Более строго, в научной литературе рассматриваемая здесь схема называется *полусумматором* (half adder), но для нас это сейчас не важно.

компьютера, должны представлять себе вид маленьких вытянутых плат такой памяти, на каждой из которых расположено несколько (обычно восемь) интегральных схем основной памяти.

Ясно, что скорость работы интегральной схемы напрямую зависит от частоты прихода тактовых импульсов, называемой *тактовой частотой* схемы. Тактовые импульсы (электрические сигналы прямоугольной формы) подаёт на электронные схемы специальный тактовый генератор (не надо путать это с тактом работы в машине фон Неймана, там это выполнение одной команды). У современных ЭВМ тактовые импульсы приходят на схемы основной (оперативной) памяти с частотой несколько сотен миллионов раз в секунду, а на схемы центрального процессора – ещё примерно в 10 раз чаще. Это должно дать Вам представление о скорости работы электронных компонент современных ЭВМ.

Теперь Вы должны почувствовать разницу в видении, например, процессора, системным программистом (на внутреннем уровне), от видения того же процессора инженером-конструктором ЭВМ. Для системного программиста архитектура центрального процессора представляется в виде устройств УУ и АЛУ, имеющих регистры, обменивающихся командами и числами с основной памятью, выполняющим последовательность команд программы по определённым правилам и т.д. В то же время, для инженера-конструктора процессор представляется в виде сложной структуры из интегральных схем, состоящих из узлов-вентилей, соединённых проводниками (в математике такая

структура называется графом – это любой набор узлов, соединённых дугами.). По этой структуре распространяются электрические сигналы, которые в дискретные моменты времени (при приходе тактовых импульсов) заставляют вентили выполнять логические операции.

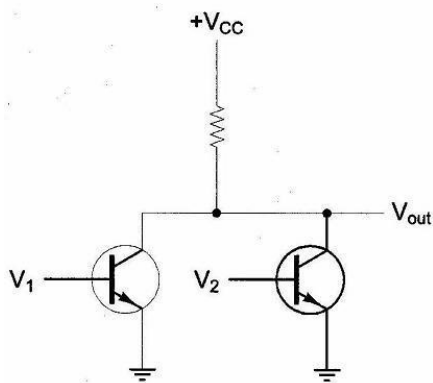


Рис. 2.3. Радиотехнический вентиль "не или".

Итак, совсем немного был рассмотрен *инженерный* уровень (уровень логических схем) архитектуры ЭВМ. Разумеется, можно и дальше продолжать спуск по этим уровням, например, на рис. 2.3 показана простейшая *радиотехническая* асинхронная схема состоящая из двух транзисторов, она имеет два входа V_1 и V_2 и один выход V_{out} и функционирует аналогично вентилю **nor** ("не или", стрелка Пирса), т.е. $V_{out} := \text{not}(V_1 \text{ or } V_2)$ (без комментариев!).

Вопросы и упражнения

Скажи мне – и я забуду, покажи мне – и я запомню, дай мне сделать – и я пойму.

Конфуций, V век до н.э.

1. Почему машина фон Неймана является *абстрактной* ЭВМ ?
2. В чём заключается принцип линейности и однородности памяти ?
3. Объясните разницу между понятиями *ячейка*, *адрес* и *машинное слово*.
4. Чем отличаются статическая и динамическая оперативная память компьютера ?
5. Сформулируйте принцип неразличимости команд и данных.
6. Как машина определяет, что находится в ячейке памяти: число или команда ?
7. Что такое язык машины ?
8. Чем отличается регистровая и оперативная память компьютера ?
9. В чём различие между регистром адреса и счетчиком адреса ?
10. В чём заключается принцип хранимой программы ?
11. Что такое вентиль и интегральная схема ?
12. Что такое тактовые импульсы и для чего они нужны ?
13. Что определяет тактовая частота ?
14. В чём разница между синхронными и комбинированными логическими схемами ?
15. Чем уровень электронных схем отличается от внутреннего уровня видения архитектуры ?

ⁱ Динамическая память DRAM на транзисторах была изобретена в 1968 году сотрудником фирмы IBM Робертом Деннардом (Robert H. Dennard). Первая микросхема DRAM Intel 1103 объёмом 1 кБит выпущена в 1970 году. Сейчас это основной вид RAM в компьютерах, она обеспечивает скорость обмена данными порядка 25-30 Гб/сек.

Каждый бит динамической памяти хранится в виде заряда в микроконденсаторе. В качестве микроконденсатора используется специально увеличенная ёмкость так называемого p-n перехода между подложкой и источником (source) транзистора, она порядка 0.01 пикофарады, что при разности потенциалов около вольта составляет примерно 10000 электронов. Зарядка и разрядка этого микроконденсатора производится с помощью подачи напряжения соответствующего знака на затвор транзистора. При чтении бита затвор транзистора открывается и, если конденсатор разряжается, то память хранила бит "0", иначе бит "1" (так уж оказалось удобнее инженерам 😊). Ясно, что при этом информация теряется, и для бита "0" приходится опять заряжать микроконденсатор. Отметим, микросхема DRAM устроена так, что за одно обращение читается не один бит, а целая строка, обычно из 32-х бит (чтение одного бита невозможно!), а так как память состоит из независимых микросхем-банков, то из неё сразу читается 16 или 32 байта (для 4-х и 8-ми банков соответственно).

Понятно, что при такой малой ёмкости микроконденсатор и "сам по себе" разряжается со временем, а "разряженный" конденсатор, наоборот, начинает накапливать заряд. Даже для весьма большого сопротивления изоляции затвора транзистора время саморазряда такого микроконденсатора составляет несколько десятков миллисекунд. Исходя из этого приходится каждые 16-64 мсек. восстанавливать, или, как говорят, регенерировать (т.е. читать и тут же записывать назад), содержимое этой памяти (это приходится делать чаще при повышении температуры). Увеличивать ёмкость микроконденсатора не целесообразно, так как возрастёт время его зарядки и разрядки через затвор транзистора и скорость работы памяти упадёт. С другой стороны, и уменьшать далее эту ёмкость нельзя, так как тогда время саморазряда станет катастрофически малым и память не сможет работать. Отметим, что при работе DRAM всё время расходует энергию, как для операций чтения-записи данных, так и для своей регенерации.

Статическая память SRAM устроена по другому. Прототипом статической памяти была память на электромеханических реле, после подачи электрического сигнала такое реле переключалось из одной позиции в другую и могло находиться там сколько угодно долго, практически не потребляя для этого энергии. Далее была статическая память на электронных лампах, по тому времени очень быстрая, но энергоёмкая и ненадёжная. Статическая память на транзисторах SRAM была запатентована сотрудником фирмы Fairchild Робертом Нормэнном в 1963 году, первый 8-битный чип создан в 1965 году. Первая 64 кБитная схема такой памяти для ЭВМ Staу-1 (по тому времени это был суперкомпьютер) появилась только в 1976 году.

Для хранения бита такая память использует специальную электронную схему из примерно 6-8 транзисторов, обычно называемую *триггером* (flір-flор), сам триггер содержит всего 2 транзистора, ещё 4-6 нужно для реализации операций чтения/записи. Триггер может находиться в двух устойчивых состояниях, которые и соответствуют значению хранимого бита. Так как никакого микроконденсатора там нет, то при хранении бита энергия не тратится, а для переключения триггера из одного состояния в другое её потребляется совсем мало.

Скорость работы динамической памяти ограничена временем заряда (и разряда) микроконденсатора, что составляет, несмотря на его крошечную ёмкость, несколько наносекунд. В то же время скорость работы статической памяти ограничена только временем прохождения много меньшего электрического заряда через затворы транзисторов триггера, что составляет уже доли наносекунды. Это позволяет статической памяти работать с той же скоростью что и вентилям в схемах центрального процессора. В частности, на статической памяти реализованы регистры и быстродействующая так называемая кэш-память, она будет изучаться в другой главе.

Энергонезависимая (Nonvolatile) память способна хранить данные и при отключении электрического питания, это, например, уже упоминавшаяся ранее постоянная память ROM. Обычно каждый бит в памяти ROM хранится в виде электрического заряда в маленькой, надёжно изолированной области, электрический заряд в эту область помещается при изготовлении такой памяти путём пережигания перемычек в электронной схеме, поэтому запись туда невозможна. Благодаря очень высокому (до 10^{14} ом) сопротивлению изоляции заряд в такой области хранится порядка 10 лет.

Далее, часто нужна энергонезависимая память, которая допускает как чтение, так и запись данных. Это, например, широко используемая так называемая флэш-память. Каждый бит в такой памяти тоже хранится в микроконденсаторе, который сделан в виде так называемого транзистора с плавающим затвором. В отличие от динамической памяти, плавающий затвор надёжно изолирован от управляющего транзистора. При чтении данных ток через транзистор зависит от наличия или отсутствия заряда в микроконденсаторе, но сам этот заряд при этом не изменяется.

При записи данных иногда требуется изменить заряд микроконденсатора на противоположный. Для заряда и разряда изолированного микроконденсатора используется *тоннельный эффект*, когда электроны, подчиняясь квантовым законам, могут "просочиться" через изолятор. При зарядке микроконденсатора электрон, "перепрыгнув" через изолирующий слой, теряет часть своей энергии и оказывается "заперт" внутри микроконденсатора. Такая запись бита путём зарядки и разрядки микроконденсатора, естественно, занимают больше времени,

чем считывание бита и, вообще говоря, требуют более высокого (10-20 вольт) электрического напряжения. Из-за высокого напряжения каждая операция записи немного повреждает изолирующий слой, поэтому число таких операций ограничено (обычно порядка сотен тысяч раз). Отметим, что плавающий затвор может хранить как один бит данных (single-level cell), так и несколько, например, четыре (multi-level cell), в зависимости от величины заряда в плавающем затворе.

При разработке флэш-памяти приходится выбирать толщину (и, следовательно, величину сопротивления) изолирующего слоя. Тонкая изоляция увеличит скорость записи данных, но уменьшит время саморазряда микроконденсатора. В энергонезависимой памяти для хранения BIOS данные хранятся около 10 лет, но и время записи в ячейку такой памяти (при "перепрошивке") может составить около 1 мс. Иногда такую память называют EEROM (электрически стираемая память). А вот современная флэш-память при отключении электрического питания хранит данные всего около двух лет, но допускает достаточно быструю запись (и не по одной ячейке, а сразу целыми блоками). Следовательно, время от времени рекомендуется вставлять флэшку во включенный компьютер, позволяя её контроллеру восстановить потерянный заряд в микроконденсаторах.

Сейчас разрабатываются и новые виды энергонезависимой памяти. В качестве примера можно привести память STT-MRAM (Spin-Torque-Transfer Magnetoresistive Random-Access Memory), каждый бит в такой памяти хранится не в виде электрического заряда, а в виде изменяемого *сопротивления* самой ячейки памяти. Ячейка такой памяти состоит из управляющего транзистора и пары ферромагнитных слоёв с тонкой прослойкой диэлектрика между ними. Один ферромагнитный слой является постоянным магнитом, а другой может менять направление намагниченности под воздействием внешнего магнитного поля, при этом используется эффект так называемого магнитного туннельного перехода MTJ (Magnetic Tunnel Junction). Такая память хранит данные примерно столько же времени, как и обычный жёсткий магнитный диск (т.е. о-чень долго).

Другие виды энергонезависимой памяти на магнитных и оптических дисках здесь рассматриваться не будут.

ⁱⁱ На 7 нм. микросхемах, например, построены смартфоны iPhone с процессором Apple A12 (модели XR, XS и XS Max), а также процессоры фирмы AMD Vega 20 и Zen 2. Отметим, что стоимость разработки микросхемы на кристалле с технологией 7 нм. составляет около 150 млн. долларов, а стоимость завода по производству таких микросхем порядка 7 млрд. долл. В 2017 году компании IBM, Samsung и GlobalFoundries представили совместный экспериментальный процессор, сделанный по 5 нм. технологии, на кристалле площадью 50 кв.мм размещены 30 млрд. транзисторов. Стоимость проектирования кристалла по 5 нм. технологии уже 400 млн. долларов, а по 3 нм. технологии – 650 млн. долларов. В 2020 году фирмы Samsung и TSMC планирует выпустить пробные экземпляры так называемых finFET транзисторов по технологии уже 3 нм.

Необходимо, однако, учесть, что уменьшение проектных норм по большей части чисто маркетинговый ход, так как при уменьшении размера транзистора расстояние между самими транзисторами на схеме (из-за тепловыделения) сокращается мало. Например, при переходе с 14 нм. к 7 нм. плотность (число транзисторов на единицу площади) практически не увеличилась, так как шаг транзисторов на схеме по-прежнему около 40 нм.! Критики отмечают, что на микросхемах с нормами 5 и 3 нм. нет ни одного элемента с такими размерами!

Заметим, что процессоры, которые вынуждены работать в условиях повышенной радиации, раньше изготавливались по значительно большим технологическим нормам. Например, в марсоходе Curiosity работал процессор RAD750, сделан по 250 нм. технологии. Компьютер на его основе выдерживает излучение до 10^5 (сам процессор до 10^6 рад, подводит память, а для человека смертельная доза около 600 рад). Сейчас, однако, разрабатываются радиационно стойкие микросхемы по проектным нормам 64, 20 и даже 14 нм., они снабжаются мощными средствами коррекции ошибок.

ⁱⁱⁱ В современных компьютерах обычно реализуют более сложные схемы, которые выполняют суммирование много-разрядных целых чисел за два или даже за один такт. При выполнении операции за один такт *внутренние* вентили схемы работают непрерывно (асинхронно), всё время после прихода первого тактового импульса (который приходит на *входные* схемы), преобразуя входные сигналы в выходные. Такие схемы называются комбинированными (синхронно-асинхронными). В нашем примере сумматора асинхронно может работать внутренний вентиль **not**. При этом необходимо точно вычислять задержки в прохождении сигналов внутри схемы, чтобы на все вентили их входные сигналы приходили в нужное время. Второй тактовый импульс приходит на выходные вентили схемы, снимая результат. Это позволяет вентилям схемы выполнять несколько операций за один такт, учитывая, что время такта процессора сейчас (на частоте 3 Гг) около 0.3 нс=300 пс., а время срабатывания вентилей порядка 50 пс. Например, уже в процессоре Pentium-IV АЛУ работало на удвоенной частоте, выполняя на вентилеях по две (простых) операции за такт. Комбинированные схемы быстрее синхронных, но "капризны" к колебаниям температуры и напряжения питания.