Глава 2. Машина Фон Неймана

В конце 40-х годов прошлого века во многих научных центрах велись разработки электронных вычислительных машин [3,26]. Можно отметить первую ламповую ЭВМ ENIAC, построенную в 1945 году сотрудниками Пенсильванского университета США Дж. Эккертом и Дж. Моучли, разработанную в Киеве под руководством академика С.А. Лебедева в 1951 году ЭВМ МЭСМ и другие проекты.


Джон фон Нейманн (1903-1957)

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

После войны Фон Нейман работал профессором Принстонского университета США (заметим, что в это же самое время и тоже профессором там работал сам А. Эйнтштейн). В 1946 году Фон Нейман (с соавторами) описал в техническом докладе конкретную ЭВМ, обладающую рядом новых особенностей [2]. 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*. Эта машина является *абстрактной моделью* ЭВМ, однако, эта абстракция отличается от абстрактных исполнителей алгоритмов (например, от хорошо известной машины Тьюринга). Машина Тьюринга может обрабатывать входные данные любого объема, поэтому этот исполнитель алгоритма принципиально нельзя реализовать. Машина Фон Неймана не поддаётся реализации по другой причине: многие детали в архитектуре этого вычислителя, как он описывается в учебниках, *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 ви́дения 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.

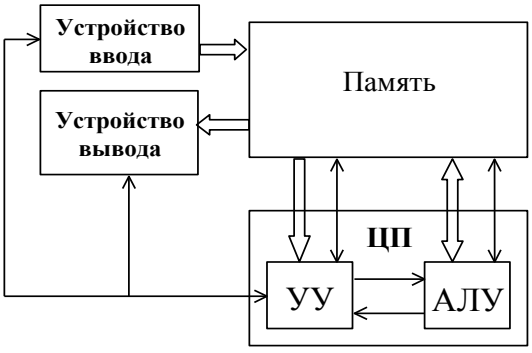


Рис. 2.1. Схема машины фон Неймана.

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, однако эти отличия удобно изучать именно как *отличия*, проводя сравнения и сопоставления с машиной Фон Неймана. При рассмотрении данной машины тоже часто будет обращаться внимание на отличия архитектуры машины Фон Неймана от современных ЭВМ. Основополагающие свойства архитектуры машины Фон Неймана будут сформулированы в ви-

де **принципов Фон Неймана**. Эти принципы многие годы определяли основные черты архитектуры нескольких первых *поколений* ЭВМ [3].[1]

На рис. 2.1 приведена схема машины Фон Неймана, как она изображается в большинстве учебников, 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 приведенного рисунка, машина Фон Неймана состоит из памяти, устройств ввода/вывода и *центрального процессора* (ЦП). Заметим, что начальный период развития вычислительной техники они назывались просто процессорами, однако позже, когда в составе ЭВМ, кроме основного, появились и другие процессоры, этот основной процессор стали называть центральным процессором (CPU – central processing unit). Центральный процессор, в свою очередь, состоит из *устройства управления* (УУ) и *арифметико-логического устройства* (АЛУ). 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*.

2.1. Память

Принцип линейности и однородности памяти. *Память* машины Фон Неймана – это линейная (упорядоченная) и однородная последовательность некоторых элементов, называемых *ячейками*. В любую ячейку памяти другие устройства машины (по толстым стрелкам на схеме рис. 2.1) могут записать и считать 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). Разумеется, время чтения из ячейки памяти может не совпадать со временем записи в неё. Такая память в современных компьютерах называется *памятью с произвольным доступом* (Random Access Memory – RAM). На практике современные ЭВМ могут иметь участки памяти разных видов. Например, некоторые области памяти поддерживают только чтение информации (по-английски эта память называется Read Only Memory – ROM), данные в такую память записываются один раз при изготовлении этой памяти, они сохраняются при отключении электрического напряжения. Другие области памяти могут допускать запись, но за бóльшее время, чем в остальную память (это так называемая *полупостоянная* память) и др.

Ячейки памяти в машине Фон Неймана нумеруются от нуля до некоторого положительного числа `N` (это и означает, что память *линейная*), причет число `N` в "настоящих" ЭВМ часто является степенью двойки, минус единица. *Адресом ячейки* называется её номер. Каждая ячейка состоит из более мелких частей, именуемых *разрядами* и нумеруемых также от нуля и до определенного числа.

---

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

[2] Рассматривается несколько 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 АЛУ, которое уже непосредственно 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 модифицированная схема близка к современным реализациям, когда внешние устройства подключаются напрямую к памяти через каналы ввода/вывода, минуя центральный процессор.

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

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*. 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 схеме (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 память старое содержимое ячейки 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 трудоемким. Дело в том, что в такой памяти (она называется *динамической* памятью (Dynamic RAM – DRAM), её использование экономически более выгодно, она дешевле) при чтении оригинал разрушается, и его приходится каждый раз восстанавливать после чтения данных. Кроме того, хранимые в динамической памяти данные разрушаются и сами по себе с течением времени (микроконденсаторы ячеек теряют электрический заряд), поэтому приходится часто (через каждые несколько миллисекунд) восстанавливать (регенерировать) содержимое этой памяти.

В компьютерах может использоваться и другой вид памяти, которая называется *статической* памятью (Static RAM – SRAM), при чтении из неё и при хранении данные не разрушаются (пока подается электрическое напряжение). Статическая память работает быстрее, чем динамическая, однако она более дорогая, т.к. требует для своей реализации больше электронных схем (1-2 транзистора на бит для динамической памяти и от 4 до 6 транзисторов на бит для статической). Память современных ЭВМ работает весьма надежно, ошибка обмена данными случается примерно раз в 10 лет.

Приведём типичные *характеристики памяти* современных ЭВМ.

1. Объем памяти – несколько 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 этой же ячейки памяти) – порядка 1-2 наносекунд (1 секунда = 10^9 нс). Следует отметить, что для упомянутой выше динамической памяти время цикла *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 всего около 30 сантиметров (а в медных проводах ещё меньше, порядка 22 сантиметров). Следовательно, быстрые компьютеры просто не могут быть слишком большими, при таком времени доступа расстояние от центрального 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 памати стоят совсем близко от коробочки центрального процессора. Вероятно, конструкторы ЭВМ с тоской вспоминают время, когда одна машина занимала большой зал со шкафами, набитыми электроникой. Впрочем, современные супер-ЭВМ опять 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, сейчас примерная стоимость оперативной памяти составляет несколько центов за мегабайт.

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.

Из неразличимости команд и данных вытекает следствие – **принцип хранимой программы**. Этот принцип является очень важным, его суть состоит в том, что программа хранится в памяти вместе с числами. Чтобы понять важность этого принципа рассмотрим, а как программы вообще появляются в памяти машины. Понятно, что, во-первых, команды программы могут, наравне с числами, вводится 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, вообще говоря, 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.[1] 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 счета самой этой программы. В частности, такая программа может *самомодифицироваться*, то есть изменять себя, во время своего счета. В настоящее время самомодифицирующиеся программы применяются крайне редко, в то время как на первых ЭВМ, как Вы увидите далее на примере одной из учебных машин, использование самомодифицирующихся программ часто было единственным способом реализации некоторых алгоритмов на языке машины.

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.[2]

2.2. Устройство Управления

Как ясно из самого 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 управляющих сигналов 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 для цифровых вычислительных машин ещё называют 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. Пример устройства, которое может выполнять команды, как и ЭВМ, но не в автоматическом режиме – обычный (непрограммируемый) калькулятор. Последовательность команд для калькулятора задаёт сам человек.

Программа – непустое множество записанных в памяти (не обязательно последовательно) машинных команд, задающих действия, описывающих шаги работы алгоритма. Команды программы обрабатывают 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 различных кодов операций, в машинном языке младшей

---

[1] В частности, 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 [20].

[2] 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, хотя по внешнему виду команды в такой памяти тоже неотличимы от чисел.

модели 8086 фирмы Intel 135 команд, а машинный язык современных наиболее распространённых компьютеров этой фирмы содержит более трёхсот различных кодов операций.

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 ячейки, расположенной в памяти непосредственно вслед за той ячейкой, в которой хранится только что выполненная команда. 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

В архитектуре машины Фон Неймана арифметико-логическое устройство (АЛУ) может выполнить следующие действия.

1. Считать содержимое некоторой ячейки памяти (машинное слово), т.е. поместить копию этого машинного слова в некоторую другую ячейку, расположенную в самом АЛУ. Если ячейки памяти расположены не в основной памяти, а в других устройствах ЭВМ, то они называются *регистровой памятью* или просто *регистрами.* Таким образом, АЛУ может читать машинное слово из памяти на один из своих регистров.
2. Записать машинное слово в некоторую ячейку памяти – поместить копию содержимого одного из своих регистров в эту 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 называется `Accumulator` и сокращается до буквы `A`, с этим обозначением Вы столкнетесь далее при изучении языка Ассемблера.

2.4. Взаимодействие УУ и АЛУ

Революционность идей Фон Неймана заключалась 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. 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 по определенным правилам адрес *следующей* команды [1] (первая букву в сокращении слова регистр в дальнейшем изложении обычно обозначается латинской буквой `R`).

Рассмотрим, например, схему выполнения команды, реализующей оператор присваивания с операцией 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

[1] Этот следующий адрес может далее быть изменён самой выполняемой командой, такие команды называются командами переходов.

такая команда есть в языке машины). После получения из памяти этой команды на регистр команд РК, 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.[1] 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 := ПАМ[x]; R2 := ПАМ[y]; S: = R1+R2; ПАМ[z] := S;
```

В дальнейшем конструкция ПАМ[A], как это принято в научной литературе по архитектуре ЭВМ, будет обозначаться как <A>, тогда схемы выполнения команды перепишется так:

```
R1: = <x>; R2: = <y>; S: = R1+R2; <z> := S;
```

Рассмотренный подробный способ выполнения ЭВМ машинных команд принято относить к так называемому уровню *микроархитектуры* компьютера [23]. 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 := <RA>; считать из ячейки памяти с адресом из регистра RA команду на регистр команд RK.
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 "забывает", какую именно команду она только что выполнила. 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. Исключение, пожалуй, составляют 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. Для экономии памяти современные компьютеры могут приписывать такой тэг не каждой ячейке в отдельности, а сразу целой последовательности ячеек, называемой сегментом. Таким образом, различают, например, сегменты команд и данных, при этом выполнение команды из сегмента данных может трактоваться как ошибка. Так нарушается принцип неразличимости команд и чисел. В такой архитектуре при попытке выполнить число как команду, либо складывать команды как числа, центральным процессором может быть зафиксирована аварийная ситуация. Очевидно, что это усложняет архитектуру ЭВМ, но позволяет повысить надежность программирования на языке машины, не допуская, как часто говорят, 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

[1] 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.

одним, а двумя числами; либо ячейки памяти могут вообще не иметь адресов, такая память называется *ассоциативной* [1] и т.д.

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

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, безусловно, 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 изменяется (например, "вдруг" не появляются новые машинные команды) и т.д.[2] 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 конце этой книги.

В заключение этого краткого 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 практически 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 на основе принципов Фон Неймана, называлась EDSAC (Electronic Delay Storage Automatic Calculator – автоматический вычислитель с электронной памятью на линиях задержки) [3] [3]. Компьютер EDSAC был построен в 1949 году в Англии 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).

И в заключение этого раздела рассмотрим совсем немного архитектуру ЭВМ на уровне инженера-конструктора. Это будет сделано исключительно для того, чтобы снять тот покров таинственности с работы центрального 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.

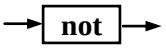
---

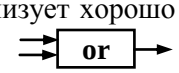
[1] Поиск нужной ячейки в ассоциативной памяти производится не по её адресу, а по содержимому хранящегося в этой ячейки машинного слова (например, считать из памяти машинное слово, хранящее целое число, кратное пяти, слово, начинающееся с целочисленного ключа 1234 и т.п.).

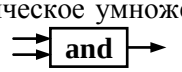
[2] В теории алгоритмов аналогом компьютера, нарушающего эти принципы, является, например, такая экзотическая модификация машины Тьюринга, у которой во время работы могут появляться и исчезать строки и столбцы её таблицы, а также изменяться содержимое клеток этой таблицы.

[3] Динамическая оперативная память этой ЭВМ была построена на так называемых ртутных линиях задержки – это длинные металлические трубки, наполненные парами ртути. 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.

Аппаратура современных ЭВМ, если не учитывать оперативную память, конструируется из некоторых относительно простых элементов, называемых в русскоязычной литературе *вентилями* (по-английски – *circuits*). Каждый вентиль является достаточно простой (электронной) схемой и реализует одну из логических операций, у него есть один или два *входа* (аргументы операции) и один *выход* (результат). На входах и выходе могут быть электрические сигналы двух видов: низкое напряжение (трактуется как ноль или логическое значение **`false`**) и высокое (ему соответствует единица или логическое значение **`true`**)[1]. Обычно напряжение менее 1 вольта трактуется как логический ноль, а напряжение более 2 вольт – как логическая единица. Можно выбрать такие основные вентили.

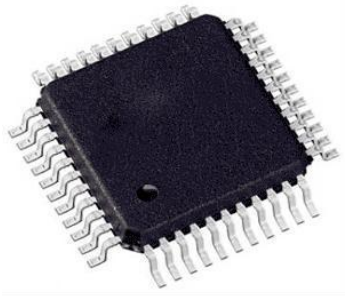
1. *Отрицание*, этот вентиль имеет один вход и один выход, он реализует хорошо известную Вам операцию отрицания **`not`** (НЕ) языка Паскаль. Другими словами, если на вход такого вентиля подается импульс высокого напряжения (значение **`true`**), то на выходе получится низкое напряжение (значение **`false`**)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

Заметим также, что чисто с технической точки зрения легче создавать вентили, задающие логические функции **`not and`** и **`not or`** (в их электронных схемах требуется на один транзистор меньше, чем для вентилей **`and`** и **`or`**), но здесь такие детали не представляют интереса. Доказано, что любую логическую функцию можно преобразовать к эквивалентному виду, в котором используются только три перечисленные выше логические операции (впрочем, можно обойтись и всего двумя упомянутыми выше операциями **`not and`** и **`not or`**).

Далее, можно считать, что 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*. Заметим, что по этому принципу работают ЭВМ, которые называются *дискретными*, в отличие от *аналоговых* компьютеров, схемы в которых работают непрерывно (всё время). Подавляющее число современных ЭВМ являются дискретными, только они и изучаются в этой книге, о принципах работы аналоговых ЭВМ немного рассказывается в последней главе данной книги. Более подробно об этом можно прочесть в книгах [1,3].


Интегральная микросхема

Из вентилей строятся так называемые *интегральные схемы* (*chips*)[2] – это набор вентилей, соединенных проводами и такими радиотехническими элементами, как сопротивления, конденсаторы и индуктивности, знакомые Вам из курса физики. Надо также сказать, что в современных интегральных схемах роль сопротивлений, конденсаторов и индуктивностей тоже выполняют транзисторы, так что схема становится однородной, т.е. состоит только из транзисторов и проводников. Каждая интегральная схема тоже имеет свои входы и выходы (их называют внешними *контактами* схемы) и реализует какую-нибудь функцию узла компьютера. Сейчас в специальной литературе интегральные схемы, которые содержат порядка 1000 вентилей, сейчас называются малыми интегральными схемами (МИС), порядка 10000 вентилей – средними (СИС), порядка 100000 – большими (БИС), а число вентилей в сверхбольших интегральных схемах (СБИС) исчисляется уже десятками и сотнями миллионов.

Интегральная схема может иметь от нескольких десятков до нескольких сотен внешних контактов. Большинство современных интегральных схем собираются на одной небольшой (прямоугольной) пластинке полупроводника с размерами порядка сантиметра. Под микроскопом такая пластинка СБИС похожа на рельефный план большого города, со своими кварталами, домами, широкими проспектами и более узкими улицами. Это многоэтажный город, он содержит до 10 изолированных друг от друга этажей-слоев со своими схемами и проводниками.

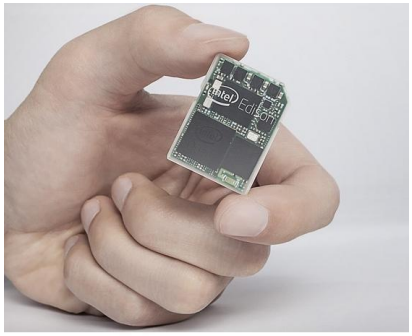
---

[1] 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.

[2] Первые интегральные схемы были созданы в 1958 году Джеком Килби (Jack Kilby) и одним из основателей фирмы Intel Робертом Нойсом (Robert Norton Noyce), на маленькой кремниевой пластинке размещалось несколько десятков транзисторов.

Впрочем, выпускаются и совсем миниатюрные ИС, например, фирма Freescale Semiconducnor 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-MO, 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, одежды.

В несколько большем формате, размером с карту флэш памяти, производятся уже "почти настоящие" ЭВМ. Например, в 2014 году фирма Intel начала производство компьютера Intel Edson для компактных и носимых устройств. Компьютер оснащен двухядерным 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. 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.).



Компьютер Intel Edison, 2014 г.

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

Дадим представление о количестве вентилей в различных электронных устройствах. Например, для того, чтобы реализовать схему самых простых электронных часов, необходимо порядка 1000 вентилей. Так, первый 4-хразрядный процессор на одной микросхеме Intel 4004 выпуска 1970 содержал 2300 транзисторов. Из 10000 вентилей уже можно собрать простейший центральный процессор (например, популярный микропроцессор 8051, встраиваемый во всевозможные устройства от светофора до холодильника, содержит около 60000 транзисторов). Центральные процессоры современных персональных ЭВМ реализуются на интегральной схеме, состоящей уже из десятков миллионов вентилей. Так, интегральная схема процессора Pentium IV содержала 42 миллионов транзисторов (это около 15 миллионов вентилей). Сейчас на одной пластинке кремния стали часто создавать не один, а несколько почти независимых друг от друга процессоров, собранные на базе таких интегральных схем компьютеры называются многоядерными. В 2010 году фирмой Intel был создан микропроцессор Itanium 9300 Tukwila, который содержал уже более 2 миллиардов транзисторов.

Важной характеристикой интегральной схемы является так называемая проектная норма (technology node, feature size) – минимальная ширина проводника на схеме. Сейчас начали производиться схемы с нормой 14 нанометров (нм), что составляет по ширине всего примерно 150 атомов. Для сравнения – размер самого маленького вируса составляет около 20 нм, что, в свою очередь, в 1000 раз меньше самого тонкого человеческого волоса. Ширина изолирующего диоксидного слоя в затворе транзистора на современных микросхемах составляет около десяти атомов. Ясно, что дальнейшее уменьшение размеров скоро натолкнётся на физический предел, за которым определяющими становятся законы квантовой физики.

В качестве примера сейчас будет рассмотрена очень простая интегральная схема сумматора,[2] которая реализует функцию сложения двух одноразрядных двоичных целых 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`), формирующимися как результат сложения `x+y`. Если трактовать значения входных переменных `x` и `y` как логические значения (0 как **`false`** и 1 как **`true`**), то можно считать, что схема реализует некоторую

[1] Чтобы наглядно представить себе такую степень чистоты, вообразите, что Вы стоите на пляже, который 10 метров в ширину, 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] Более строго, в научной литературе рассматриваемая здесь схема называется *полусумматором*, но для простоты изложения это не будет приниматься во внимание.

логическую функцию с двумя аргументами и двумя логическими результатами. Запишем таблицу истинности для этой логической 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<>y = x xor y = (x or y) and not(x and y)
b = x and y
```

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

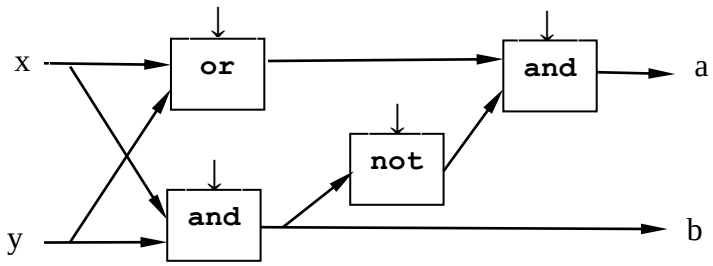


Рис. 2.2. а) Сборка двоичного сумматора из вентилей, ↓ – тактовые импульсы.

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 вентилей), два контакта для подачи электрического питания (ясно, что без энергии ничего работать не будет) и, возможно, другие контакты. Суммирование чисел x и y в приведенной выше схеме осуществляется после последовательного прихода трёх тактовых импульсов (как говорят, за три такта). Современные компьютеры обычно реализуют более сложные схемы, которые выполняют суммирование многоразрядных целых чисел за два или даже за один такт.[1]

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

Заметим, что основная память современных компьютеров также реализуется в виде набора нескольких сверхбольших интегральных схем. Пользователи, заглядывавшие внутрь персонального компьютера, должны представлять себе вид маленьких вытянутых плат такой памяти, на каждой из которых расположено несколько (обычно восемь) интегральных схем основной памяти.

[1] При выполнении схемой операции за один такт её вентили работают непрерывно, всё время после прихода тактового импульса преобразуя входные сигналы в выходные. При этом необходимо очень точно вычислять задержки в прохождении сигналов внутри схемы, чтобы на все вентили их входные сигналы приходили в нужное время. Это позволяет выполнять несколько операций за один такт.

Ясно, что скорость работы интегральной схемы напрямую зависит от частоты прихода тактовых импульсов, называемой *тактовой частотой* схемы (не надо путать её с тактом работы, т.е. выполнением одной команды, в машине Фон Неймана). 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 ви́дении, например, центрального процессора, системным программистом (на внутреннем уровне), от ви́дения 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 центральный процессор представляется в виде сложной структуры из интегральных схем, состоящих из узлов-вентилей, соединенных проводниками и такими радиотехническими элементами, как сопротивлениями, конденсаторами и индуктивностями (В математике такая структура называется графом – это любой набор узлов, соединенных дугами.). По этой структуре распространяются электрические сигналы, которые в дискретные моменты времени (при приходе тактовых импульсов) заставляют вентили выполнять логические операции.

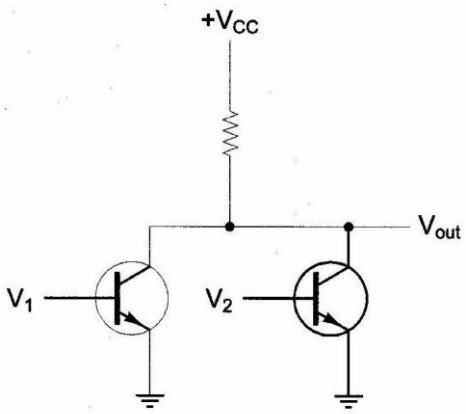


Рис. 2.3. Радиотехническая схема вентиля "не или".

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* схема из двух транзисторов, которая имеет два входа V_1 и V_2 и один выход V_{out} и функционирует аналогично вентилю "не или" V_{out}`:=`**`not`**`(`V_1 **`or`** V_2`)` (без комментариев!).

Вопросы и упражнения

1. Почему машина Фон Неймана является *абстрактной* ЭВМ?
2. В чём заключается принцип линейности и однородности памяти?
3. Объясните разницу между понятиями *ячейка*, *адрес ячейки* и *машинное слово*.
4. Чем отличаются статическая и динамическая память компьютера?
5. Сформулируйте принцип неразличимости команд и данных.
6. Что такое язык машины?
7. Чем отличается регистровая и основная память компьютера?
8. В чём различие между регистром адреса и счетчиком адреса?
9. В чём заключается принцип хранимой программы?
10. Что такое вентиль и интегральная схема?
11. Что такое тактовая частота?