Глава 2. Машина фон Неймана

Если вы точно назовёте, чего не может делать машина, я всегда смогу создать машину, которая сможет именно это.
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), МЭСМ (Малую Электронную Счетную Машину), созданную в 1951 году в Киеве под руководством академика С.А. Лебедева 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.

По мнению многих современников, фон Нейман был в числе самых выдающихся учёных прошлого века. Приехав в Америку, в 1933 году он начал работать в Принстонском Институте Перспективных Исследований США. Достаточно упомянуть, что там в это время работали А. Эйнштейн и К. Гёдель. Один его коллега шутил: «Джонни не совсем человек, но так долго живёт среди людей, что научился хорошо на них походить».

Ещё в 1945 году, ознакомившись с ЭВМ ENIAC, фон Нейман принял участие в проектировании нового компьютера с названием EDVAC (Electronic Discrete Variable Automatic Computer). Вскоре в соавторстве с Артуром Берксом (Arthur Burks) и Германом Голдстайном (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*.

Документ, описывающий проект EDVAC, первоначально не предназначался для публикации в открытой печати. Первым с этим документом познакомился Г. Голдстайн (Гольдштейн), математик по образованию, он был куратором проекта от Баллистической лаборатории Армии США, которая финансировала разработку данной машины. Вскоре он разослал ксерокопии этой работы для ознакомления 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»].[1] Недоброжелатели потом говорили, что фон Нейман был не автором, а 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 [2] 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 Академии Наук и Академии Искусств и Наук США 🙂.


Электронная лампа

Сама машина EDVAC была реализована в 1950 году, она работала на тактовой частоте 1 МГц (примерно 1000 операций в секунду), имела 32 Кб оперативной памяти и содержала около 3000 электронных ламп.

---

[1] Русский перевод «Беркс А., Голдстейн Г., Нейман Дж. Предварительное рассмотрение логической конструкции электронного вычислительного устройства // Кибернетический сборник. М.: Мир, 1964. Вып. 9»

[2] Заметим, что UNIVAC была первой серийной ЭВМ 1951 года выпуска, она имела намять из 1000 ячеек по 12 *десятичных* разрядов и стоила 1 млн. долларов 🙂. Было продано 46 таких машин. Первая отечесвенная серийная ЭВМ Стрела начала выпускаться в 1953 году.

Машина фон Неймана является *абстрактной моделью* ЭВМ, и в этом смысле она похожа на абстрактные исполнители алгоритмов, например, на машину Тьюринга, однако между ними есть и существенное различие. 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 описана не на внутреннем, а только на концептуальном уровне ви́дения архитектуры.

В некотором смысле машина фон Неймана подобна *абстрактным структурам данных*. У таких 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*, проводя сравнения и противопоставления с машиной фон Неймана. Мы тоже часто будет 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].


С.А. Лебедев (1902-1974)

Как уже упоминалось, в послевоенные годы в СССР под руководством выдающегося советского учёного Сергея Алексеевича Лебедева проводились работы по созданию первых отечественных ЭВМ. В 1954 году С.А. Лебедев возглавил московский Институт Точной Механики и Вычислительной техники (ИТМиВТ), где были созданы многие из первых отечественных ЭВМ. При этом С.А. Лебедев, независимо от фон Неймана, сформулировал аналогичные и даже более детальные принципы построения вычислительных машин. Однако, из-за секретности таких работ в СССР эти принципы не были в то время опубликованы в открытой печати и не стали достоянием широкой общественности.

На рис. 2.1 приведена схема машины фон Неймана. На этом рисунке толстыми (двойными) стрелками показаны *потоки команд и данных*, а тонкими – передача между отдельными устройствами компьютера *управляющих* и *информационных сигналов*.[1] 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 о том, что они успешно выполнили действия, предписанные управляющими сигналами, либо зафиксировали ошибки в своей работе.

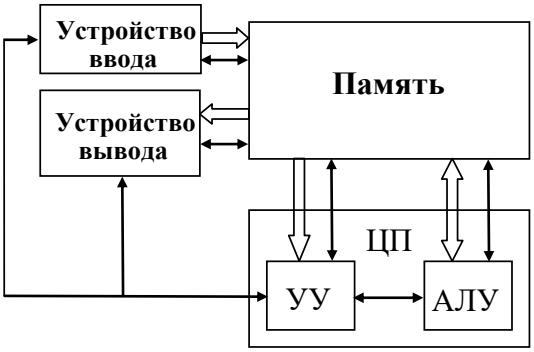
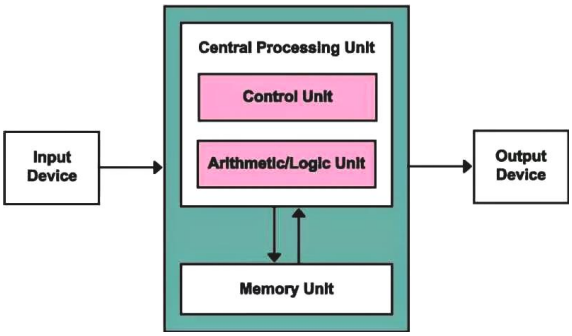


Рис. 2.1. Схема машины фон Неймана.

[1] 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, но сейчас нам это не важно.

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, так что в оригинале была такая схема:



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 через специальные контроллеры, минуя центральный процессор. Это называется прямой доступ к памяти (DMA – direct memory access).

Как видно из приведённого рисунка, машина фон Неймана состоит из памяти (memory – этот термин впервые введён фон Нейманом!),[1] устройств ввода/вывода и центрального процессора (ЦП).

Заметим, что в начальный период развития вычислительной техники центральный процессор 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 (multicore) ЭВМ сейчас обычно вместо процессоров говорят о процессорных ядрах, а термин «центральный процессор» практически не используется.

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

Сейчас будут последовательно рассмотрены все узлы машины фон Неймана и выполняемые ими функции. Основные понятия и принципы фон Неймана при этом выделяются жёлтым фоном и подчёркиваются.

2.1. Память

Память определяет быстродействие.
Джон фон Нейман

Записывая что-то в память ЭВМ, помните, куда Вы это положили.
Первая аксиома Лео Бейзера

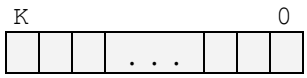
Принцип линейности и однородности памяти. Память машины фон Неймана – это линейная (упорядоченная) и однородная последовательность некоторых элементов, называемых ячейками (memory sells). В любую ячейку памяти другие устройства машины (по толстым стрелкам на рис. 2.1) могут записывать и считывать информацию, 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 *оперативной памятью* или *памятью с произвольным доступом* RAM (Random Access Memory).

[1] В первых отечественных ЭВМ память называлась «накопитель», так как в те времена кибернетика считалась в СССР лженаукой, и ЭВМ не могла обладать памятью, которая есть только у живых существ.

000	ячейка
001	ячейка
002	ячейка
003	ячейка
004	ячейка
...	ячейка
N	ячейка

Оперативная память (main storage) современных ЭВМ не совсем однородна. Сначала она достаточно медленно находит месторасположение нужного байта, зато потом может быстро считать или записать много (например, 64) байт подряд. Отметим, что так же устроена и память на магнитых и твердотельных (SSD) дисках. Кроме того, современные ЭВМ могут иметь участки (оперативной) 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 бóльшее время, чем в остальную память (это так называемая *полупостоянная* память, на которой обычно реализованы схемы для хранения служебных программ, например, BIOS), флэш-память (она допускает стирание не по одному байту, как BIOS, а только сразу целого блока) и др.

Ячейки памяти в машине фон Неймана нумеруются от нуля до некоторого положительного числа N, это и означает, что память *линейная*, т.е. упорядоченная, за каждой ячейкой (кроме последней) есть следующая, и перед каждой ячейкой (кроме первой) есть предыдущая. Число N в «настоящих» ЭВМ часто равно степени двойки, минус единица, т.е. $N=2^q-1$. Адресом ячейки называется её номер.[1] Каждая ячейка состоит из более мелких частей, именуемых разрядами, они тоже нумеруемых от нуля и до определенного числа K, причём обычно нумерация идёт справо-налево:



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. Сам фон Нейман тоже был сторонником принципа использования двоичной системы счисления, что позволяло хорошо описывать архитектуру узлов ЭВМ с помощью логических (булевских) выражений. Большинство современных ЭВМ имеют ячейки памяти размером один байт (8 бит).

Говорят, что когда-то пираты Вест-индских островов для совершения мелких платежей (выпивая в тавернах 🙂) разрубали серебряный пиастр и золотой дублон на 8 равных частей, которые и назывались *битами* (bit – кусочек). В 1948 году математик Джон Уайлдер Таки (John Wilder Tukey) вместо словосочетания *binary digit* (двоичная цифра), стал применять короткое bit. Сейчас восемь последовательных бит составляют один *байт*, но сначала байты состояли из 6 или 7 бит, первый байт из 8 бит появился в семействе ЭВМ IBM System/360 только в 1964 году.

Впервые в Европе двоичная система счисления была описана в 1703 году знаменитым Готфридом Лейбницем в работе «Объяснение двоичной арифметики (Explication de l'Arithmétique Binaire)». При этом сам Лейбниц ссылается на древнекитайскую так называемую «книгу Перемен», где были описаны 64 гексаграммы, соответствующие записям первых 64 чисел в 4-битной (т.е. шестнадцатеричной) системе счисления.

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.

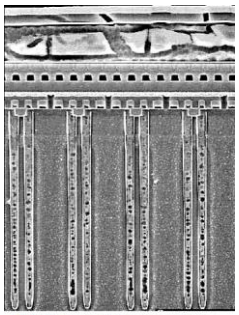
То, что за один раз можно обменяться с памятью только одним машинным словом, является узким местом машины фон Неймана, как говорят, «бутылочным горлышком» (bottleneck). Ситуация усугубляется тем, что большая часть потока через эту узость составляют не полезные данные, а адреса этих данных и другая служебная информация. Например, чтобы прочитать из памяти целое *число* длиной 4 байта, надо сначала послать в память *адрес* этого числа (4 или 8 байт) и некоторые управ-

[1] Сейчас это называется *физическим* адресом ячейки памяти. Процессоры современных компьютеров работают также с *логическими* и *виртуальными* адресами, это мы будем обсуждать позже.

ляющие данные. Чтобы исправить эту ситуацию память современных ЭВМ позволяет за один раз по начальному адресу читать и писать сразу большие блоки данных (по 32, 64 и более байт).

Ячейки памяти располагаются не только в самой (основной) памяти, но и в других устройствах (УУ, АЛУ и т.д.), в этом случае они называются регистрами, а сама основная память называется при этом оперативным запоминающим устройством (ОЗУ). Таким образом, при чтении копия машинного слова читается из ячейки ОЗУ и записывается на регистр некоторого устройства, а при записи копия машинного слова из некоторого регистра посылается в ячейку ОЗУ. Кроме того, вообще говоря, копия машинного слова из одного регистра может записываться и в другой регистр.

Заметим, что на практике решение задачи сохранения исходного машинного слова при чтении из ячейки оперативной памяти является нетривиальным и достаточно трудоёмким. Дело в том, что в такой памяти, которая называется динамической памятью с произвольным доступом DRAM (Dynamic Random Access Memory), при чтении оригинал разрушается (стирается), и его приходится каждый раз восстанавливать после чтения данных. Использование этой памяти экономически более выгодно, она дешевле. Биты в этой памяти хранятся в крошечных, «канавочных» конденсаторах (capacitor), похожих на пары глубоких вертикальных траншей (см. фото слева). Такие конденсаторы имеют очень маленькую ёмкость и быстро теряют электрический заряд. Хранимые в динамической памяти данные с течением времени разрушаются и сами по себе, поэтому приходится часто (примерно каждые 15-30 миллисекунд) восстанавливать содержимое этой памяти. Ничего страшного в этом нет, сейчас при работе микросхемы динамической памяти тратят на эту регенерацию примерно 1% своего времени.


Микроконденсаторы DRAM, 20 нм

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

Приведём типичные *характеристики памяти* современных (персональных) 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 (а в оптоволокне вообще 20 сантиметров!). 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.


Платка DRAM памяти

Заглянув внутрь системного блока персонального компьютера можно заметить, что продолговатые планки оперативной памяти стоят совсем близко от коробочки процессора (на которой обычно расположен вентилятор).[1] 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,

[1] На графических картах такая память не вставляется в разъём материнской платы, а впаивается прямо в графическую карту.

тем она, естественно, дороже. Конкретные значения стоимости памяти меняются весьма быстро с развитием вычислительной техники.

Принцип неразличимости команд и данных.

Знание некоторых принципов легко возмещает незнание некоторых фактов.
Клод Адриан Гельвеций

С точки зрения программиста машинное слово представляет собой либо команду, либо подлежащее обработке данное (число, символьная информация, элемент изображения и т.д.). Для краткости в дальнейшем будем называть такие данные «числами». Этот 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. Некоторые из современных компьютеров отступают от принципа неразличимости команд и данных, это, например, отечественные процессоры Эльбрус.

Следующим является принцип хранимой программы. Этот принцип является очень важным, его суть состоит в том, что программа хранится вместе с числами в одной и той же памяти. 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 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 алгоритма. 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.

Принцип хранимой программы 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 (mercury delay lines) – длинные металлические трубки, наполненные парами ртути, в этих парах распространялись звуковые волны кодирующие нули и единицы. Когда волна достигала конца такой трубки, она детектировалась (как ноль или единица) и тут же генерировалась с начала трубки.

По длине трубки укладывалось несколько сотен длин волн (бит), скорость доступа к отдельному биту была примерно 200 мкс. Заметим, что эта память не совсем отвечала принципу однородности памяти фон Неймана, так как одинаковое время доступа к каждому биту было только *в среднем* за большое число обращений. В дальнейшем Фредерик Уильямс разработал более перспективную (однородную) 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 обычно ЭВМ имеют общую память для команд и чисел.

Следствием принципов хранимой программы и неразличимости команд и данных 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 [ii] [см. сноску в конце главы]. В настоящее время самомодифицирующиеся машинные программы (self-modifying code) применяются редко, например, это любят делать так называемые полиморфные (самоизменяющиеся) вирусы. В то же время на первых ЭВМ, как Вы увидите далее, использование самомодифицирующихся 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. В то же время и в современных ЭВМ в особой быстродействующей памяти типа кэш (см. разд. 14.1) команды и данные храняться в разных частях такой памяти.

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.

2.2. Устройство Управления

Власть – палка о двух концах.
Фрэнк Хербер. «Дюна»

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.

Обстоятельства переменчивы, принципы никогда.
Оноре де Бальзак

Этот принцип имеет ещё и другое название –принцип 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 непосредственно задаёт сам человек.

Программа – множество записанных в памяти (не обязательно последовательно) машинных команд, задающих действия, описывающих шаги работы алгоритма.[1] Команды программы обрабатывают данные, хранимые в памяти компьютера и на регистрах. 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 около 1.5 тысяч.[2]

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. Сейчас говорят, что команды программы сериализованы (serialized). 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 (с бóльшим номером) 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, для машины Тьюринга это например, чтения из клетки ленты символа, которого нет в заголовке ни одной колонки таблицы.

---

[1] «Компьютерная программа – комбинация компьютерных инструкций и данных, позволяющая аппаратному обеспечению вычислительной системы выполнять вычисления или функции управления (стандарт ISO/IEC/IEEE 24765:2010)».

[2] Не надо пугаться, бóльшая часть этих команд предназначена для весьма специфических векторных и мультимедийных операций, обычно их «знают» только компиляторы.

2.3. Арифметико-логическое устройство

Лучший способ в чём-то разобраться 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 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, АЛУ может 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` (`S:=R1+R2`). В англоязычной литературе этот регистр часто называется `Accumulator` и сокращается до буквы `A`, с этим обозначением Вы столкнетесь далее при изучении языка Ассемблера.

2.4. Взаимодействие УУ и АЛУ

Компьютерная программа делает то, что Вы приказали ей сделать, а не то, что Вы хотели, чтобы она сделала.

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 адреса (или *счётчиком адреса,* в англоязычной литературе его часто обозначают `IP` – instruction pointer). *Перед* выполнением текущей команды УУ записывает в него адрес *следующей* команды [2] (первая букву 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 выполнить следующие действия, обычно называемые микрооперациями):

---

[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 быть далее изменён самой выполняемой командой, такие команды, как уже говорилось, называются командами переходов.

```
1. R1 := ПАМ[x]
2. R2 := ПАМ[y]
3. S  := R1+R2
4. ПАМ[z] := S
```

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. В дальнейшем конструкция `ПАМ[A]`, как это принято в компьютерной литературе, будет обозначаться как `<A>`, тогда схема выполнения команды

```
<z> := <x>+<y>
```

перепишется так:

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

Итак, каждая машинная команда (операция) разлагается на микрооперации, которые последовательно выполняются. Сейчас такие микрооперации называются в англоязычной литературе microops, m-ops, uops, µops (читается мю-опс), а в русскоязычной литературе *мопами*.

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 схему работы машины фон Неймана:

```
repeat
    RK := <RA>;  считать из ячейки с адресом RA команду на регистр команд RK.
    RA := RA+1;  увеличить счётчик адреса на единицу.
    Выполнить команду, хранящуюся в регистре RK.
until STOP or ERROR;
```

Один шаг такого цикла (выполнение одной команды – instruction execution cycle) называется тактом работы машины фон Неймана. 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 биты со специальным признаком, называемый тэгом (tag), который и определяет, чем является это машинное слово [iii] [см. сноску в конце главы].

Практически все современные ЭВМ нарушают принцип однородности и линейности памяти. Память может, например, состоять из двух частей со своей независимой нумерацией ячеек в каждой части (с такой архитектурой мы вскоре познакомимся); или быть двумерной, когда адрес ячейки задается не одним, а двумя числами; либо ячейки памяти могут вообще не иметь адресов, такая память называется *ассоциативной*. Поиск нужной ячейки в ассоциативной памяти (content addressed memory) производится не по её *адресу* (которого нет), а по *содержимому* хранящегося в этой ячейке машинного слова (на-

пример, считать из памяти первое машинное слово, хранящее целое число, кратное пяти; или слово, начинающееся с цифр 1234 и т.п.).[1]

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, 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 в работе самого фон Неймана явно не описаны, 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, число регистров в устройствах. Далее, считалось, что машинный язык при выполнении программы не изменяется (например, «вдруг» не появляются новые машинные команды) и т.д. 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 (флешка). 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 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 программирования, 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»). Его описание очень простое, буквально на одной странице (можно посмотреть в Интернете). Людям-программистам пока не удалось написать на нём ни одной программы, все известные программы на языке Malbolge получены другими программами (написанными на традиционных языках).

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, так называемый *контекст программы* (текущее состояние регистров компьютера, сведения об открытых файлах и т.д.). В дальнейшем в любой момент возможно возобновление счёта 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.

Компьютер EDSAC, 1949 г.

Первая ЭВМ, построенная по принципам фон Неймана, называлась EDSAC (Electronic Delay Storage Automatic Calculator – автоматический вычислитель с электронной памятью на линиях задержки) [3]. Этот компьютер 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 на такто-

[1] Можно отметить, что такой принцип доступа к информации широко распространён. Например, все записи в базах данных снабжаются уникальныит *ключами*, по которым (а вовсе не по адресам этих записей в памяти компьютера) эти записи и ищутся.

вой частоте 500 Кгц (14000 сложений или 100 умножений в секунду). Ввод производился с перфоленты, вывод на пишущую машинку, память всего 225 байт. Заметим, что именно от этой машины принято отсчитывать **первое** поколение ЭВМ (все предшествующие «не совсем настоящие» компьютеры можно условно отнести к нулевому поколению).

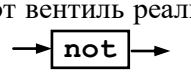
В конце этого раздела рассмотрим совсем немного архитектуру ЭВМ на уровне электронных логических схем (на уровне инженера-конструктора). Это будет сделано исключительно для того, чтобы снять тот покров таинственности с работы процессора, который есть сейчас у некоторых читателей: как же машина может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

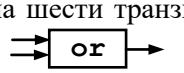
Аппаратура современных ЭВМ, если не учитывать оперативную память, конструируется из некоторых относительно простых элементов, чаще всего называемых в русскоязычной литературе (логическими) вентилями (по-английски – *digital logic gates*).

Русский «сантехнический» термин *вентиль* здесь больше подходит, чем английский *gate* (безликие «врата», правда с «затвором» 🙂). Так, кран-смеситель, к которому подводится холодная и горячая вода, ведёт себя аналогично вентилю **`or`**. С другой стороны, однако, «кран» **`and`** ведёт себя непонятно, а **`not`** вообще мистическим образом: из него течёт вода, когда она к нему не подводится 🙄.

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 (трактуется как ноль или логическое значение **`false`**) и высокое (ему соответствует единица или логическое значение **`true`**).[1] Обычно для современных микропроцессоров с электрическим питанием около 0.7 вольта, напряжение менее 0.1 вольта трактуется как логический ноль, а напряжение более 0.4 вольта – как логическая единица.[2] «Плохие» значения входных напряжений приводят к неустойчивой работе вентилей, поэтому все ЭВМ содержат специальные схемы для стабилизации напряжения, иначе в процессоре возникает аварийная ситуация. Отметим, что при таком маленьком напряжении для работы микросхемы нужет большой ток (порядка 100 ампер, типичная мощность микропроцессора около 65 ватт).

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. Обычно этот вентиль реализуется на двух транзисторах. Далее в схемах этот вентиль изображается так: 

2. *Дизъюнкция* или логическое сложение, этот вентиль реализует хорошо известную Вам операцию Паскаля **`or`** (ИЛИ). Обычно этот вентиль реализуется на шести транзисторах, будем изображать его как 

3. И, наконец, вентиль, реализующий *конъюнкцию* или логическое умножение, в Паскале это операция **`and`** (И). Обычно этот вентиль реализуется на шести транзисторах, будем изображать его как 

Любую логическую функцию можно преобразовать к эквивалентному виду, в котором используются только три перечисленные выше логические операции. Такой набор операций называется *базисом*.

Здесь стоит заметить, что с чисто технической точки зрения легче создавать вентили, задающие логические операции **`not and`** (**`nand`** – штрих Шеффера, обозначается как вертикальная палочка |) и **`not or`** (**`nor`** – стрелка Пирса, обозначается как ↓), в их электронных схемах требуется на два

---

[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. Такие устройства называются оптоэлектронными. Бывают и магнитные (так называемые спиновые) вентили.

[2] Сначала напряжение питания транзисторов быдо 15 вольт, потом стало 5, потом 2.5 и так до 0.9 вольта, скажем, что для кремнеевых транзисторов предел напряжения около 0.5 вольта, ниже него перестают работать.

транзистора меньше, чем для вентилей **and** и **or** (да и работают они быстрее). С помощью и **nand** и **nor** можно выразить все логические операции (это ещё два базиса, на основании которых можно построить все электронные схемы). Таким образом, все электронные схемы (да и константы **true** и **false**) можно построить только на одном типе вентилей, например, для **nand**:

```
not X ≡ X | X;  true ≡ X | (X | X);
X or Y ≡ (X | X) | (Y | Y);
X and Y ≡ (X | Y) | (X | Y);  и т.д.
```

Интегральная микросхема

Из вентилей строятся так называемые интегральные схемы (*integrated circuits, chips*) – это набор вентилей, соединенных проводами и такими радиотехническими элементами, как сопротивления, конденсаторы и индуктивности, знакомыми Вам из курса физики. В современных интегральных схемах, однако, роль сопротивлений, конденсаторов и индуктивностей часто тоже выполняют транзисторы (таких транзисторов на схеме где то 20-30%), 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 вентилей, называются малыми интегральными схемами (МИС), порядка 10000 вентилей – средними (СИС), порядка 100000 – большими (БИС), а число вентилей в сверхбольших интегральных схемах (СБИС) исчисляется миллионами и миллиардами.

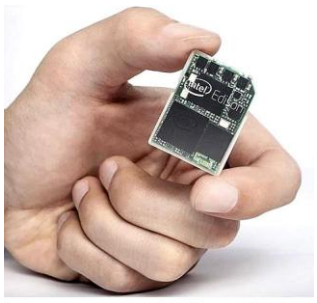
Первые интегральные схемы были созданы в 1958 году Джеком Килби (Jack Kilby) из компании Texas Instruments и одним из основателей фирмы Intel Робертом Нойсом (Robert Norton Noyce). На маленькой кремниевой пластинке размером 11.1x1.6 мм размещалось несколько десятков транзисторов (сам транзистор был изобретён ещё в 1947 году). Первая отечественная микросхема была создана в 1961 году в Таганрогском Радиотехническом Институте на основе технологии, разработанной в НИИ «Пульсар».

Интегральная схема может иметь от нескольких десятков до нескольких тысяч внешних контактов. Например, процессор Intel Core i7 имеет на нижней поверхности 1366 контактов, расположенных с шагом около 1 мм., а 80-ядерный ARM процессор Ampere Ultra (2020 год, 7 нм.) – 4926 контактов.

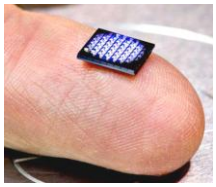
Большинство современных интегральных схем собираются на одной небольшой (прямоугольной) пластинке полупроводника с размерами порядка нескольких сантиметров. Такая пластинка СБИС похожа на рельефный план большого города, со своими кварталами, домами, широкими проспектами и более узкими улицами. Это многоэтажный город, он может содержать до 20 изолированных друг от друга этажей-слоев со своими схемами и проводниками.


В объёме пары игральных костей уместится 6400 чипов Freescale Kinetis KL02

Дешёвые встроенные 32-битовые процессоры невысокой производительности размещаются на кристалле площадью около 0.1 см^2, а микропроцессоры, встраиваемые в различные устройства (принтеры, холодильники, микроволновки и т.д.), часто располагаются на площади меньше 0.03 см^2. 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-MO, 4 Кб оперативной и 32 Кб 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.


Компьютер Intel Edison, 2014 г.

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 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. Всё работает под управлением 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.).



Самые маленькие специализированные компьютеры имеют сейчас размеры 1 и даже 0.3 мм (см. фото слева). 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 самого предмета бесцветной токопроводящей краской.

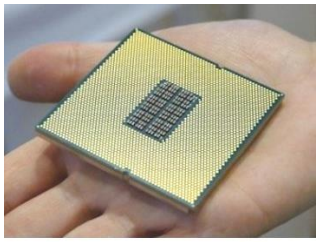
Обычно в качестве основы в ИС используется очень чистый кремний, сейчас в нём допускается 1 атом примеси на 10^{10} атомов кремния. Это чистота 0.1 ppt (parts per million – частей на миллиард), так называемая чистота «десять девяток». Чтобы наглядно представить себе такую степень чистоты, вообразите, что Вы стоите на морском пляже шириной 100 метров, который простирается вдаль на 10 километров и весь покрыт белым песком. Так вот, на поверхности всего этого пляжа Вы сможете найти только одну чёрную песчинку примеси. Ясно, что для получения такого чистого кремния чистота вспомогательных реагентов (воды, кислот и т.д.) должна быть ещё на один-два порядка выше.

Процессор Intel 8086, 1978 г.

Дадим представление о количестве вентилей в различных электронных устройствах. Скажем, для того, чтобы реализовать схему самых простых электронных часов, необходимо порядка 1000 вентилей. Например, первый в мире 4-разрядный микроконтроллер Intel 4004 выпуска 1971 года содержал 2250 транзисторов. Из 10000 вентилей 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. 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.

Примерно в 1989 году процессоры стали содержать более миллиона транзисторов на кристалле. 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 (без увеличения площади кристалла).[1] Примерно с 2015 года, однако, закон Мура резко «замедлился» и сейчас удвоение происходит уже через 20 лет! В настоящее время число транзисторов на больших интегральных схемах составляет порядка 10 миллиардов. Это, например, 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, например, уже поверхность ЦП Intel Pentium 4 при работе выделяла около 30 Вт/см^2. Для сравнения, сковородка, на которой жарится яичница, выделяет примерно 10 Вт/см^2 🙂, а у современных процесссоров тепловыделение практически такое же, как в сопле стартующей ракеты. Впрочем, современные процессоры могут сохранять работоспособность до температуры примерно в 100 градусов. Выделяемая микропроцессором мощность (которую необходимо отводить) называется его термопакетом TDP (Thermal Design Power), обычно это от 15 до 350 и более ватт. От 20 до 50% этой мощности уходит на паразитные токи утечки в микросхеме, они пропорциональны *четвёртой* (!) степени температуры микросхемы [iv] [см. сноску в конце главы].


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 (сам транзистор значительно больше). Так, первый процессор Intel 8086 1978 года был создан по 3000-нанометровой (нм.) технологии, а знаменитый процессор Intel Pentium 1993 года выпуска производился по 800 нм. технологии и содержал 3.1 млн. транзисторов.

В настоящее время для ЭВМ общего назначения в основном произво-

[1] Вместе с законом Мура (между прочим, это основатель фирмы Intel ⚠) 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.

дятся процессоры с нормой 14 нм., но для смартфонов есть микросхемы с нормой 10, 7, 5 и даже 3 нм. [v] [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. Так как размер атома кремния составляет 0.24 нм., то 10 нм. это примерно 80 атомных слоёв.[1] Для сравнения, размер самого маленького вируса около 30 нм. (знаменитый короновирус 100 нм.).[2] Таким образом, на одном квадратном миллиметре помещается около 100 млн. транзисторов, на срезе человеческого волоса около 1.5 миллиона, а на острие иглы примерно 50 тыс. транзисторов.

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. Одна микросхема изготавливается несколько месяцев и стоит около 1 млн. долларов. Общее число транзисторов примерно $1.2\ 10^{12}$. На основе этой микросхемы создан компьютер Cerebras CS-1 (жидкостное охлаждение, потребляемая мощность около 20 Квт), он предназначен для работы в больших центрах обработки данных в сфере искусственного интеллекта. Самым большим сейчас по-видимому является процессор Cerebras WSE-2 2020 года выпуска. На пластине размеров 220x220 мм. он содержит 850 тыс. ядер (по техпроцессу 7 нм) и $2.6*10^{12}$ транзисторов. Потребляемая мощность 15 кВт. Может поддерживать работу систем искусственного интеллекта со 120 10^{12} параметрами (см. разд. 15.5).

Современные интегральные схемы (чипы), без преувеличения, являются самыми сложными устройствами, когда-либо производимыми человеком. Завод по производству таких чипов по сложности превосходит и большой адронный коллайдер, и (будущий) термоядерный реактор. Стоимость одного такого завода превышает 10 млрд. долларов, а некоторые стоящие на этом заводе «станки» стоят более 100 млн. долларов, а есть и такие важные «станки», которые сейчас производятся только в одном месте на Земле (это, например, нидерландская фирма Holdings NV ASML). Например, «станок» TWINSCAN NXE:3600D для выполнению литографии для чипов по техпроцессу 5 нм. весит 180 тонн, состоит из 100 тыс. деталей и стоит 180 млн. долл. Кремниевые пластины для чипов сейчас изготавливаются только в пяти фирмах. Производство сложного чипа содержит около 700 шагов и занимает более трёх месяцев ⚠. При уменьшении технологической нормы стоимость разработки чипа многократно возрастает. Так, стоимость разработки чипа на 28 нм. составляет 50 млн. долларов, а на 3 нм. уже миллиард долларов. На 2021 год выпускать микросхемы по 180 нм. технологии могли 29 фирм в мире (2 российские), а по 7 нм. технологии уже только три! А вот для 2 нм. фактически остаётся только одна тайваньская фирма TSMC (Taiwan Semiconductor Manufacturing Company) на оборудовании нидерландской фирмы ASML (Advanced Semiconductor Materials Lithography).

К сожалению, в обычный оптический микроскоп увидеть мелкие детали микросхемы не удаётся. Типичный размер отдельных элементов массовых микросхем сейчас составляет 14 нм. и менее, а длина волны фиолетового цвета (самого коротковолнового из различимых глазом человека) – около 380 нм. Теоретический предел разрешения (так называемый дифракционный предел Аббе) составляет примерно ½ длины волны, так что посмотреть на транзистор микросхемы «глазами» не получится ☹.[3] Образно говоря в оптический микроскоп можно различить не отдельные транзисторы-дома, а только целые большие «микрорайоны». Современная технология для получения рисунка на кристалле использует ультрафиолетовое излучение с длиной волны 13.5 нм, это на самой границе между жёстким ультрафиолетом и уже *рентгеновским* диапазоном [vi] [см. сноску в конце главы].

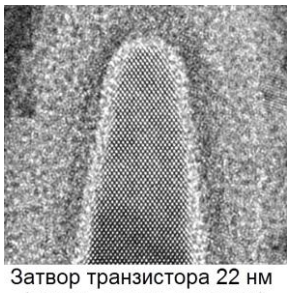
---

[1] Шаг в кристаллической решётке кремния 0.543 нм.=5.43 Å (1 нм=10 Å – ангстрем, типично атомная единица длины), однако кристаллическая решётка кремния не кубическая, в так называемая гранецентрированная (как у алмаза!), так что на расстоянии в 10 нм. укладываются не 40, а примерно 80 атомных слоёв.

[2] Длина самой молекулы ДНК вируса во много раз больше, но в вирусе она очень плотно упакована. Общая длина ДНК человека (3 099 734 149 пар оснований) имеет длину около 2 метров и тоже плотно упакована.

[3] Сейчас разработаны так называемые оптические микроскопы ближнего поля, они преодолеваютдифракционный барьер за счёт того, что получают изображение объекта на рассоянии, меньше длины волны λ (свет «не успевает размазаться») ⚠. Так можно получить разрешение $\lambda/20$, это около 13 нм.

Для получения изображения одного транзистора и его частей придётся использовать специальные атомный *сканирующий* (туннельный, зондовый) или атомный силовой микроскоп, которые особой, очень тонкой иглой «ощупывает» поверхность кристалла. Игла неподвижна, перемещается поверхность с точностью 0.001 Å, положение иглы отслеживается отражённым лазерным лучём (в зондовым микроскопе) или изменением туннельного тока (в туннельном микроскоп) между иглой и поверхностью.


Затвор транзистора 22 нм
(точки - атомы кремния)

Острие иглы имеет размер 2-3 атомов и приближается к поверхности на расстояние 0.1 нм., что позволяет различать рельеф даже отдельных атомов. В качестве примера слева показано изображение вертикального среза верхней части канала затвора так называемого FinFET транзистора (fin – «плавник», FET – field effect transistor), сделанного по 22 нм. технологии, а справа – схема такого транзистора.[1]

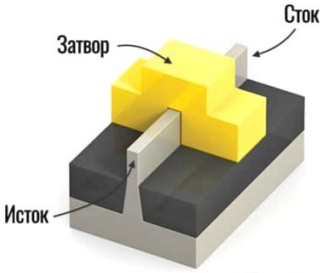
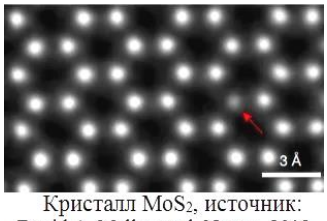


Схема транзистора FinFET

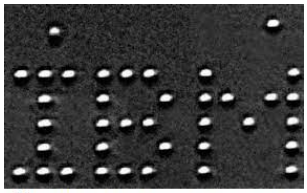
Отметим, что сейчас у транзистором может быть не один, а два-три «плавника». Ширина такого «плавника» у современных транзисторов 5-8 нм., однако длина затвора не может быть меньше примерно 20 нм., дальше вступают в силу квантовые эффекты и затвор просто перестаёт «запирать» транзистор. Для дальнейшего уменьшение этой длины необходимо перейти к другим полупроводникам (дисульфид молибдена (MoS_2), диселенид вольфрама (WSe_2), углеродные нанотрубки и др.). В этом случае длина затвора может быть снижена до примерно 1 нм. (это 1-3 молекулы), но это технология будущего.

Изображение затвора транзистора получено при максимальном разрешении атомного сканирующего микроскопа. Ширина канала транзистора («плавника») около 8 нм., а длина канала (см. схему справа) примерно 20 нм. Регулярно расположенные точки в канале являются отдельными атомами кремния в его кристаллической решётке (это проводник), по сторонам – полиморфный кремний (полупроводник). Ширина более светлого изолирующего слоя окиси кремния SiO_2 между ними около 1 нм., что составляет всего около 5 молекул SiO_2. Такая тонкая изоляция, даже учитывая, что SiO_2 почти идеальный диэлектрик, приводит к большим токам утечки, поэтому до 25% и более потребляемой микросхемой мощности бесполезно уходит в тепло.



Кристалл MoS_2, источник:
David A. Muller et al. Nature, 2018

Заметим, что сейчас используется и так называемый электронный микроскоп TEM (Transmission Ekecnron Microscope) он имеет схожее разрешение около 0.15 нм. В качестве примера слева показано полученное этим микроскопом изображение отполированной поверхности кристалла дисульфида молибдена MoS_2. Видно, как две молекулы MoS_2 образуют в кристалле правильный шестиугольник, стрелкой показан дефект кристаллической решётки – один атом серы отсутствует (1 нм.=10 Å).


Надпись атомами ксенона

Подавая на иглу такого микроскопа напряжение нужного знака, можно «схватить» отдельный атом, оторвать его от поверхности, перенести и опустить на другое место. В качестве примера слева показана надпись, сделанная атомами ксенона на поверхности охлаждённого до 4 К кристалла никеля. Правда, атомы ксенона быстро испарились с поверхности 😐.

Интересно, что и у современных жёстких дисков (HDD) размер магнитной частицы, хранящей бит, тоже около 5 нм., такова же толщина магнитного слоя, а головки чтения/записи имеют размер около 30 нм. Благодаря этому плотность записи на HDD около 7 Тбит/см^2. Головки «плавают» на воздушной (или гелиевой) подушке над поверхностью пластины на расстоянии около 5 нм. Поверхность самой пластины жёсткого диска отполирована по 14-му (самому последнему!) классу чистоты, при этом перепад высот по всей поверхности пластины примерно 5 нм. Чтобы оценить степень чистоты полировки пластины представьте, что мы увеличили размер диска так, чтобы длина дорожки составила 4000 км, тогда максимальный перепад высот на всём этом пути будет около 4 см. ⚠, а расстояние головки от поверхности диска будет около 1 метра.

Скажем также, что современная 3D NAND флэш-память также производится по 12 нм. технологии (сейчас производится 3-мерная 192-слойная память). Стоит также отметить, что и плотность за-

[1] Рисунок взят с сайта `https://deep-review.com/articles/what-is-nanometer-process`.

писи на магнитную ленту в системах резервного копирования больших объёмов данных (стримерах) составляет сейчас около 30 Гбит/кв.см. (200 Гбит/кв.дюйм). Ясно, что дальнейшее уменьшение всех этих размеров скоро натолкнётся на физический предел, за которым определяющими становятся уже законы квантовой физики.

Для упомянутой выше технологии изготовления современных микросхем с проектной нормой 14 нм. отражающие металлические линзы не должны иметь шероховатости более, чем 0.5 нм, это уже сравнимо с размером атома. Если увеличить такую линзу до размера 1000 км., то размер неровностей не должен быть больше, чем 1 мм ⚠.

Чтобы подчеркнуть, насколько близко мы подошли к физическому пределу уменьшения основных размеров электронных элементов компьютера, скажем, что сейчас секунда определяется как время 9 192 631 770 переходов между двумя сверхтонкими уровнями основного состояния атома цезия-133, таким образом, один переход соответствует 0.092 нм.=0.92 Å (это длина, соответствующая пределу точности нашего измерения времени).

В качестве примера сейчас будет рассмотрена очень простая интегральная схема сумматора, которая реализует функцию сложения двух одноразрядных двоичных чисел.[1] 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 логическую функцию с двумя аргументами и двухбитным результатом. Запишем таблицу истинности (true table) для этой функции:

x	y	b	a	x+y=ba
0	0	0	0	0+0=00
0	1	0	1	0+1=01
1	0	0	1	1+0=01
1	1	1	0	1+1=10

Можно вычислить, что выходные величины `a` и `b` будут определяться такими логическими формулами (в базисе операций **`not`**, **`and`** и **`or`**):

```
a = x<>y = x xor y = (x or y) and not(x and y)
b = x and y
```

Реализуем схему этого сумматора как набор вентилей, 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 вентилей. Заметим, что формально можно ввести отдельный вентиль «дублировать» с одним входом и двумя идентичными выходами, но не будем усложнять нашу схему.

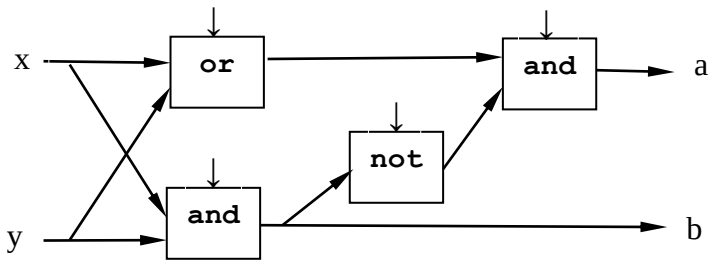


Рис. 2.2. а) Сборка двоичного сумматора из вентилей, ↓ – тактовые импульсы.

Каждый вентиль срабатывает (т.е. преобразует входные сигналы в выходные) не всё время (непрерывно), а только тогда, когда на этот вентиль по специальному управляющему проводу приходит так называемый тактовый импульс (tick of the internal clock). Тактовые импульсы (на нашей схеме

[1] Более строго, рассматриваемая здесь схема называется *полусумматором* (half adder), полный сумматор более сложный, он получает на входе складываемые биты `x` и `y`, а также бит переноса `z` (единица «в уме») от сложения двух предыдущих битов многоразрядного двоичного числа.

они изображены тоненькими стрелочками ↓) часто называются *синхронизирующими*, так как они задают моменты одновременного (синхронного) срабатывания вентилей 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). [vii] [см. сноску в конце главы]

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.

Рис. 2.2. б). Пример ИС двоич

Значения на выходах нашей схемы зависят только от значений на её входах, такие схемы называются *комбинационными* (combinational). Эти схемы «не помнят» предыдущие входные значения, у них нет памяти. Бывают и более сложные *последовательностные* (sequential) схемы, значения на выходе у них могут зависеть от нескольких предыдущих значений на входах (у них есть память).

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.

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. [viii] [см. сноску в конце главы]

Теперь Вы должны почувствовать разницу в ви́дении, например, 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 процессор представляется в виде сложной структуры из интегральных схем, состоящих из узлов-вентилей, соединенных 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.

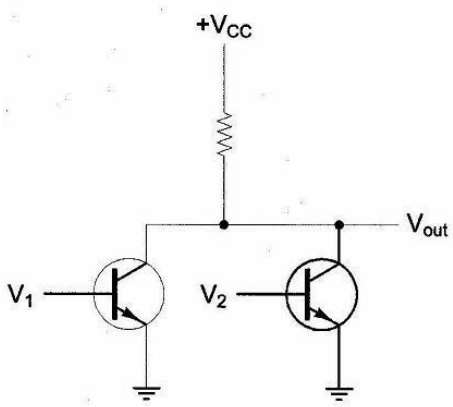


Рис. 2.3. Радиотехнический вентиль "не или".

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 как вентиль **nor**

(«не или», стрелка Пирса), т.е. $V_{out} := \mathbf{not}\,(V_1\ \mathbf{or}\ V_2)$ (без комментариев!). Обычно в микросхемах сопротивление (на схеме извилистая линия сверху) реализуется в виде двух транзисторов.

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. В чём заключается принцип хранимой программы ? Чем он отличается от принципа неразличимости команд и данных ?
8. Что такое язык машины ?
9. Чем отличается регистровая и оперативная память компьютера ?
10. В чём различие между регистром адреса и счётчиком адреса ?
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 асинхронными логическими схемами ?
15. Чем ви́дение архитектуры на уровне электронных схем отличается от внутреннего уровня ви́дения архитектуры ?

---

[i] Для продвинутых читателей.

Все жалуются на свою память, но никто не жалуется на свой разум.
Франсуа де Ларошфуко

Динамическая память DRAM (Dinamic RAM) на транзисторах была изобретена в 1966 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 Гбайт/сек. Размер одной ячейки (бита) памяти DRAM составляет около 20 нм.

Каждый бит динамической памяти хранится в виде заряда в микроконденсаторе (storage capacitor). В качестве такого конденсатора используется специально увеличенная ёмкость так называемого p-n перехода между подложкой и истоком (source) транзистора, она порядка 0.01 пикофарады (10^{-15} Ф), что при разности потенциалов около вольта составляет примерно 40000 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 "1", иначе бит "0". Ясно, что при этом хранимая информация теряется, и для бита "1" приходится опять заряжать микроконденсатор. 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 (паразитный) заряд. Даже для весьма большого сопротивления изоляции затвора транзистора (порядка 10^{12} ом) время саморазряда такого микроконденсатора составляет несколько десятков миллисекунд. Исходя из этого, приходится примерно каждые 15-30 мсек. восстанавливать, или, как говорят, регенерировать (т.е. читать и тут же записывать назад), содержимое этой памяти (это приходится делать чаще при повышении температуры микросхемы памяти). 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.

Любопытно, что при очень низких температурах DRAM может хранить данные без регенерации в течении нескольких минут. Хакеры демонстрировали, как после выключения компьютера они быстро доставали из него платки памяти, охлаждали в жидком азоте и вставляли в другой компьютер, после чего читали содержимое практически всей оперативной памяти ⚠.

Для обеспечения высокой скорости чтения/записи сейчас (особенно в памяти для графических процессоров) широко используется модификация DRAM под названием HBM (high bandwidth memory – память с высокой пропускной способностью). По этой технологии стопка (стек) чипов памяти соединяются сквозными соединениями и монтируются в одну микросхему памяти, при этом достигается ширина шины памяти 1024 бита, что для 4-х микросхем (4092 бита) обеспечнивает скорость 3-4 Тбайт/сек. Отметим, что также уменьшается размер чипа и напряжение питания, а значит повышается энергоэффективность.

Статическая память SRAM (Static RAM) 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 находится 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 формы Fairchild Робертом Нормэном в 1963 году, первый 8-битный чип создан в 1965 году. Первая 64 битная схема такой памяти для ЭВМ Cray-1 (по тому времени это был суперкомпьютер) появилась только в 1976 году.

Для хранения бита такая память использует специальную электронную схему из 6-8 транзисторов, обычно называемую *триггером* (flip-flop), 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 практически 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, порядка 5-10 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.

Оперативная память современных ЭВМ работает весьма надежно, 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 ПК массового производства 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 ECC (Error Correction Code). Эти схемы в несколько раз повышают надёжность памяти, но увеличивают её стоимость примерно на 10%.

Стоит также отметить, что оперативная память менее надёжна, чем процессор, так как содержит значительно больше электронных схем. Например, динамическая RAM объёмом 4 Гб содержит $2*(4*8*2^{30})=2^{36}$. транзисторов, а современный процессор только примерно $10^{10} \approx 2^{32}$ транзисторов. С другой стороны, оперативная память имеет значительно более регулярную структуру, чем процессор. Интересно, что, по мнению многих исследователей, подавляющее большинство ошибок оперативной памяти вызываются нейтронами от вторичных космических лучей, 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 значительно уменьшают частоту возникновения неисправленных ошибок (до, примерно, 1 в месяц на 4 Гбита DRAM).

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 остаётся в этой области при изготовлении такой памяти после 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 (floating-gate transistor). В отличие от динамической памяти, плавающий затвор (иногда его называют *карманом*) 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* или так называемая *инжекция горячих электронов*, когда электроны, подчиняясь квантовым законам, могут «просочиться» через изолятор. Для этого используется достаточно высокое напряжение (около 20 вольт). При зарядке микроконденсатора электрон, «перепрыгнув» через изолирующий слой, теряет часть своей энергии и оказывается «заперт» внутри микроконденсатора. Иногда такую память называют EEROM (электрически стираемая память). Такая запись бита путём зарядки и разрядки микроконденсатора, естественно, занимают больше времени, чем считывание (порядка микросекунды, вместо нескольких наносекунд для памяти DRAM) и требуют более высокого электрического напряжения (20 вольт вместо 1 вольта для чтения). Таким образом, флэш-память при записи примерно в 1000 раз медленнее, чем DRAM, но дешевле.

Из-за использования высокого напряжения, каждая операция записи немного повреждает изолирующий слой, поэтому число таких операций ограничено (обычно порядка 10^{10} 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. Начиная с 2010 года флэш-накопители вместо плавающего затвора, используют более эффективную (за счёт более совершенной изоляции) так называемую память с ловушкой заряда CTF (Charge Trap Flash), мы здесь описывать её работу не будем.

При разработке флэш-памяти приходится выбирать толщину (и, следовательно, величину сопротивления) изолирующего слоя. Тонкая изоляция увеличит скорость записи данных, но уменьшит время саморазряда микроконденсатора. А вот современная флэш-память при отключении электрического питания хранит данные всего около 10 лет, но допускает достаточно быструю запись (и не по одной ячейке, а сразу целыми блоками). Следовательно, время от времени рекомендуется вставлять флэшку во включённый компьютер, позволяя её контроллеру восстановить потерянный заряд в микроконденсаторах. Кроме того, контроллеру флэшки нужно время для таких операций, как сборка мусора и перераспределения страниц для их равномерного износа (эти вопросы мы рассматривать не будем).

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 электрического *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-ч-ень долго). Другие виды энергонезависимой памяти на магнитных и оптических дисках здесь рассматриваться не будут.

[ii] Для продвинутых читателей. Выполнение принципа хранимой программы (с возможной самомодификацией этой программы) очень дорого обходится для современных ЭВМ. Во-первых, отступая от принципа фон Неймана последовательной работы, они берут на выполнение сразу много (несколько десятков) команд, находящихся на разных стадиях выполнения. Тогда при изменении одной из этих команд в памяти машины все остальные должны быть отменены (попросту выброшены из процессора) и потом повторены с начала.

Во-вторых, почти все современные ЭВМ имеют по несколько процессоров (ядер), которые могут работать по одной программе. Как следствие, когда один процессор изменяет команду программы в памяти, он должен послать всем остальным процессорам сигналы прерываний, чтобы они тоже проверили свои частично выполненные команды. Как видим самомодификация программы очень накладна.

[iii] Пожалуй, впервые однобитные тэги появились в одной из первых «настоящих» ЭВМ EDVAC в 1950 году. Там первый разряд в машинном слове определял, что хранится в этом слове (0 – число, 1 – команда). Тэги длиной 3 бита в 48-битных ячейках памяти успешно использовались в ЭВМ B6500 фирмы Burroughs 1966 года выпуска. Из современных ЭВМ можно отметить отечественную Эльбрус [31], система тэгов в этой машине близка к понятию типов данных в языках программирования высокого уровня. Например, там только одна (обобщённая) команда сложения, а тег операндов определяет тип слагаемых (целые или вещественные, с одинарной или двойной точностью). Собственно, архитектура этой ЭВМ и поддерживает язык программирования высокого уровня Эль-76. В процессорах фирмы Intel вещественные регистры сопроцессора имеют теги из двух бит, описывающие содержание регистра:

a) допустимое число,
b) ноль,
c) недопустимое число («не число» `NaN`, ±∞, ненормализованное число, будут описаны далее),
d) «пустой» регистр.

На языке Free Pascal таким свойством обладают переменные особого типа `variant`, каждая такая переменная снабжается служебным полем (тэгом), в котором хранит свой текущий тип. Таковы и все переменные языка Python.

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».

[iv] Для продвинутых читателей. Для контроля температуры по всей поверхности кристалла размещены десятки температурных датчиков (в том числе по 10-15 на каждое процессорное ядро). Заметим, что общая протяженность проводников внутри микросхемы составляет порядка 10 километров, а плотность электрического тока в таком проводнике около 10^5 А/см^2. Такая плотность энергии приближается к величинам в сопле стартующей ракеты ⚠. Без охлаждения процессор «перегорит» за несколько секунд.

Любопытно, что транзисторов на чипе стало так много, что все они одновременно работать не смогут, не хватит электрической мощности. Приходится, например, на двух ядрах включать блоки векторных регистров, а два других ядра отключать 😕. Отметим, что сейчас вычислительные мощности по обработке данных потребляют около 10% всей вырабатываемой электрической энергии в мире.

Сейчас для борьбы с тепловыделением приходится окружать активно работающие участки пристала областями так называемого «тёмного кремния» (dark silicon), эти области содержат «мёртвые», отключённые от электрического питания массивы транзисторов. Только иногда эти транзисторы удаётся задействовать в схемах, работающих очень редко (в основном в кэш-памяти). Ситуация забавная, мы можем разместить на маленьком кристалле миллиарды транзисторов, но заставить их работать все одновременно не можем, кристалл просто сгорит, и не поможет даже охлаждение жидким азотом. В то же время мы знаем, что мозг человека содержит порядка 100 триллионов нервных окончаний (синапсов), но при работе потребляет всего от 10 до 30 ватт ⚠.

---

[v] Для продвинутых читателей. 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 10 млрд. долл. В 2017 году компании IBM, Samsung и Global Foundries 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 начали выпускать схемы с так называемыми finFET транзисторами по технологии уже 3 нм., стоимость такого завода фирмы TSNC на Тайване 19.5 млрд. долларов.

Необходимо, однако, учесть, что такое уменьшение проектных норм по большей части чисто маркетинговый ход. Критики отмечают, что на микросхемах с нормами 5 и 3 нм. нет ни одного элемента с такими размерами! Плотность упаковки транзисторов, однако, возрастает. Например, при переходе с 14 нм. к 10 нм. на 1 кв.мм. число транзисторов возросла с 37 до 100 млн. В 2021 году фирма IBM выпустила опытный образец микросхемы с нормой 2 нм., на 1 кв.мм помещается уже 333 млн. транзисторов. Заметим, что 2 нм. это минимальная ширина проводника, сам транзистор мно-о-го больше: 75x40 нм., причём главный размер транзистора – ширина его базы – составляет примерно 25 нм. и при уменьшении проектных норм с 28 до 5 нм. не уменьшается ⚠.

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

[vi] Для любознательных. Предыдущее поколение лазоров на основе фторида аргона давало длину волны 193 нм. Далее следовало большое окно, в котором не было подходящих лазеров с более короткой длиной волны. Основой же производства современных микросхем является использование излучения лазера с длиной волны 13.5 нм., это на самой границе между жёстким ультрафиолетом и уже *рентгеновским* диапазоном Такое излучение даёт плазма паров олова с температурой 600000 градусов, эта плазма получается испарением капелек расплавленного сверхчистого олова сверхмощными углекислотными лазерами. Эти капли диаметром примерно по 25 микрон падают со скоростью 70 м/сек. (5-10 тысяч капель в секунду) в камере с глубоким вакуумом на специальную поверхность. На этой поверхности по каждой капле надо попасть лучём лазера два раза, первый раз слабый луч просто сплющивает каплю олова, в второй 30 киловатный луч лазера испаряет каплю. Потреб-

ляемая лазерной установкой мощность около 1 Мвт! Для сравнения, чтобы резать сталь нужна в три раза меньшая мощность лазера.

Не так давно (около 2008 года) научились делать «хитрые» линзы (и, главное, фокусирующие зеркала) с так называемой отрицательным показателем преломления (отрицательной кривизной), они позволяют поднять предел разрешения примерно до ¼ длины волны. При длине волны лазера 13.5 нм. это позволяет производить микросхемы с технологической нормой около 3 и даже 2 нм. Необходимые зеркала произвоит фирма Zeiss, они состоят из примерно 100 слоёв, неровность полировки сопоставима с размером атома. В литературе приводится сравнение: если представить, что такое зеркало имеет размер в 1000 километров, то высота неровностей будет составлять всего несколько миллиметров ⚠.

Сейчас появились новое поколение фотолитографических машин High-NA EUV (стоимостью уже более 300 млн. долларов) с так называемой большой числовой апертурой (0.55 вместо 0.33 на прежних машинах, не будем здесь объяснять, что это такое). Это позволяет снизить теоретический предел разрешения до 18 Å (ангстрем, т.е. 1.8 нм.).

Несмотря на такую малую длину волны, элементы микросхемы с рамерами менее половины длины волны сильно искажаются из-за дифракции света. Все прямые углы скругляются, прямые линии получаются волнистыми и укорачиваются и т.д. Одним из методов борьбы с этим является применение так называемой коррекции эффектов оптической близости (OPC – Optical Proximity Correction). Суть этого метода заключается в том, что в исходное изображение *намеренно* вносятся такие изменения, чтобы изображение, полученное на микросхеме, снова стало чётким, без искажений. Это так называемая обратная задача, её решение требует значительных вычислительных ресурсов (много времени рассчётов на супер-ЭВМ), соответствующие алгоритмы, естественно, нигде не публикуются (являются собственностью соответствующих фирм). Ниже показана схема работы этого меода:

Коррекция оптической близости

Стоимость проектирования микросхемы с техпроцессом 3 нм. (30-50 млрд. транзисторов) составляет около 650 млн. долларов.

[vii] Для продвинутых читателей. В современных компьютерах обычно реализуют более сложные схемы, которые выполняют суммирование многоразрядных целых чисел за два или даже за один такт. Для этого можно применить два приёма.

Во-первых, *внутренние* вентили схемы могут работать непрерывно (асинхронно), всё время после прихода первого тактового импульса (который приходит на *вход* схемы, сразу на все её вентили), преобразуя входные сигналы в выходные. Такие схемы называются комбинированными (синхронно-асинронными). 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 вентиля порядка 50 пс. Например, уже в процессоре Pentium-IV АЛУ работало на удвоенной частоте, выполняя на вентилях по две (простых) операции за такт. Комбинированные схемы быстрее синхронных, но «капризны» к колебаниям температуры и напряжения питания.

Во-вторых, можно распараллелить процесс сложения многоразрядных целых чисел. Например, будем параллельно (на двух сумматорах) складывать младшие и старшие 32 бита 64-битных чисел. При этом, в свою очередь, сделаем не один, а два отдельных сумматора для старших 32-битных частей складываемых чисел, они будут получать идентичные слагаемые, но один будет предполагать, что при сложении младших частей будет получаться перенос "1" («единица в уме»), а второй будет предполагать, что будет перенос "0" («ноль в уме»).

Таким образом мы в 2 раза уменьшим время сложения 64-разрядных чисел, правда 1/3 работы (для переноса "1" или для переноса "0") будет сделана зря. Такой приём называется спекулятивным вычислением (speculative execution), когда вычисления делаются по двум (или более) ветвям, все из которых, кроме одного, окажутся ненужными.

Понятно, что, если, в свою очередь, разбить 32-битные слагаемые на 16-битные, 16-битные на 8-битные и т.д., то можно каждый раз уменьшать время вычисления суммы вдвое. Всего можно ускорить сложение в $\log_2(64)=6$ раз, правда при этом большая часть логических вентилей сумматора будет работать зря, но нам не жалко, у нас миллиарды вентилей 🙂.

[viii] Для продвинутых читателей. У современных ЭВМ разные устройства (оперативная память, процессорные ядра, кэш и т.д.) работают с разнами тактовыми частотами. Обычно есть базовый тактоый генератор,

скажем, на 100 Мгц, а у каждого устройста свой умножитель частоны, например, для современных процессорных ядер базовая частота увеличивается в 30-40 раз. Тактовые импульсы очень важны в работе цифровых схем, на их генерацию тратится около 30% общей мощности, потребляемой микросхемой ⚠. Все вентили должны быть равноудалены от источника тактовых импульсов, поэтому эти импульсы подаются в центр чипа и распределяются по сбалансированному двоичному дереву (вентили – листья этого дерева). Здесь Вам должен помочь курс предыдущего семестра «Алгоритмы и алгоритмические языки» (см., например, сайт

http://algorithm.cs.msu.ru

Отдельные части современных процессоров могут не работать значительное время выполнения программы. Это, например, устройство для работы с вещественными числами, векторная арифметика, периферийные шины и т.д. Для вре́менного отключения таких устройств для них снижают частоту подачи тактовых импульсов (это так называемый режим троттлинга – throttling), или снижают электрическое напряжение (потребляемая мощность пропорциональна квадрату напряжения). Для более длительного выключения устройств на них совсем перестают посылать тактовые импульсы иди даже отключают электрическое питание. Всем этим занимается «рачительный завхоз» чипа микроконтроллер PCU (Package Control Unit).

В многоядерных процессорах таким образом можно отключать отдельные ядра целиком, это позволяет значительно экономить электрическую энергию. Для более глубокой экономии энергии всего компьютера существуют два основных режима. В режиме standby отключаются только ядра, но работает оперативная память, а в режиме hibernate отключается и память, «в работе» остаётся только контроллер прерываний и PCU.