Оперативная память ЭВМ

Виды памяти персонального компьютера

Компьютерная память бывает двух видов: ***внутренняя*** *и* ***внешняя***. ***Внутренняя*** память состоит из ячеек, каждая из которых имеет свой уникальный адрес, или номер. Элемент информации сохраняется в памяти с назначением ему некоторого адреса. Чтобы отыскать эту информацию, компьютер находит ячейку и копирует ее содержимое. Емкость отдельной ячейки памяти называется словом.

Виды памяти персонального компьютера

Внешняя память реализуется в виде довольно разнообразных устройств хранения информации и обычно конструктивно оформляется в виде самостоятельных блоков. Назначение **внешней памяти** компьютера заключается в долговременном хранении информации любого вида. Выключение питания компьютера не приводит к очистке внешней памяти. Объем этой памяти в тысячи раз больше объема внутренней памяти. Обращение к внешней памяти требует гораздо большего времени.

Виды памяти персонального компьютера

Метод доступа:

Последовательный доступ (*sequential access memory, SAM*) — ячейки памяти выбираются (считываются) последовательно, одна за другой, в очерёдности их расположения. Вариант такой памяти — стековая память.

Произвольный доступ (*random access memory, RAM*) — вычислительное устройство может обратиться к произвольной ячейке памяти по любому адресу.

Виды памяти персонального компьютера

Доступные операции с данными:

- ***Память только для чтения*** (*read-only memory, ROM*)
- ***Память для чтения/записи***

Память на программируемых и перепрограммируемых ПЗУ (ППЗУ и ППППЗУ) не имеет общепринятого места в этой классификации. Её относят либо к подвиду памяти «только для чтения», либо выделяют в отдельный вид.
Также предлагается относить память к тому или иному виду по характерной частоте её перезаписи на практике: к ***RAM*** относить виды, в которых информация часто меняется в процессе работы, а к ***ROM*** — предназначенные для хранения

Виды памяти персонального компьютера

Удалённость и доступность для процессора:

- ***Первичная память*** (сверхоперативная, СОЗУ) — доступна процессору без какого-либо обращения к внешним устройствам.
  - ***регистры процессора*** (*процессорная* или *регистровая память*) — регистры, расположенные непосредственно в АЛУ;
  - ***кэш процессора*** — кэш, используемый процессором для уменьшения среднего времени доступа к компьютерной памяти. Разделяется на несколько уровней, различающихся скоростью и объёмом

Виды памяти персонального компьютера

- ***Вторичная память*** — доступна процессору путём прямой адресации через ***шину адреса*** (*адресуемая память*). Таким образом доступна ***оперативная память*** (память, предназначенная для хранения текущих данных и выполняемых программ) и ***порты ввода-вывода*** (специальные адреса, через обращение к которым реализовано взаимодействие с прочей аппаратурой).

Виды памяти персонального компьютера

- ***Третичная память*** — доступна только путём непрямой последовательности действий. Сюда входят все виды ***внешней памяти*** — доступной через устройства ввода-вывода. Взаимодействие с третичной памятью ведётся по определённым правилам (протоколам) и требует присутствия в памяти соответствующих программ. Программы, обеспечивающие минимально необходимое взаимодействие, помещаются в ПЗУ, входящее во вторичную память.

Классификация видов памяти

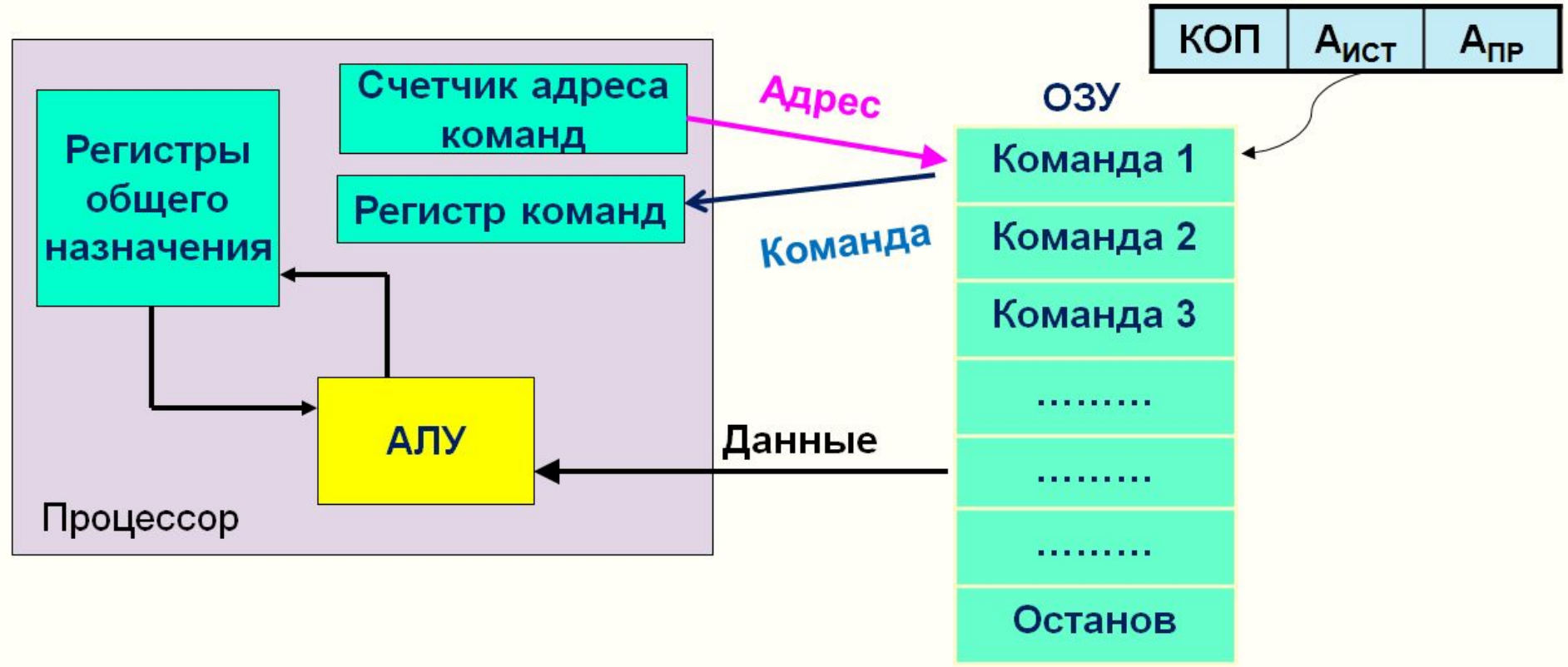
- Память
  - Энергозависимая (оперативная память или ОЗУ, кэш-память, буферная)
    - Статическая
    - Динамическая
  - Энергонезависимая
    - Только для чтения
      - ПЗУ, CD-R, DVD-R, Blue ray - диски
    - Перезаписываемая
      - Флеш-карты
      - Жёсткий диск (HDD)
      - CD-RW, DVD-RW, Blue ray - диски

Оперативная память - энергозависимая часть системы компьютерной памяти, в которой временно хранятся данные и команды, необходимые процессору для выполнения им операций.

Оперативная память является одним из трех обязательных устройств персонального компьютера: процессор, память, периферийные устройства.

Взаимодействие процессора и ОЗУ

Последовательное считывание команд из памяти и их выполнение. Номер (адрес) очередной ячейки памяти, из которой будет извлечена следующая команда программы, указывается специальным устройством – **счетчиком команд** в устройстве управления.



Содержащиеся в оперативной памяти данные доступны только тогда, когда на модули памяти подаётся напряжение, то есть, компьютер включён. Пропадание на модулях памяти питания, даже кратковременное, приводит к искажению либо полному уничтожению данных в ОЗУ. В общем случае, оперативная память содержит данные операционной системы и запущенных на выполнение программ, поэтому от объёма оперативной памяти зависит количество задач, которые одновременно может выполнять компьютер.

Обмен данными между процессором и оперативной памятью производится:

непосредственно;

через сверхбыструю память 0-го уровня — регистры в АЛУ, либо при наличии аппаратного кэша процессора — через кэш.

Энергосберегающие режимы работы материнской платы компьютера позволяют переводить его в режим *сна*, что значительно сокращает уровень потребления компьютером электроэнергии. В режиме *гибернации* питание ОЗУ отключается. В этом случае для сохранения содержимого ОЗУ операционная система (ОС) перед отключением питания записывает содержимое ОЗУ на устройство постоянного хранения данных (как правило, жёсткий диск). Например, в ОС Windows содержимое памяти сохраняется в файл hiberfil.sys, в ОС семейства Unix — на специальный swap-

Статическая и динамическая оперативная память

В настоящее время оперативную память выполняют на микросхемах.
Наибольшее распространение получили два вида ОЗУ:
микросхемы ***динамического*** (Dinamic Random Access Memory — ***DRAM***) или ***статического*** (Static Random Access Memory — ***SRAM***) типа.

- статическое (*SRAM*): память в *виде массивов триггеров*;
- динамическое (*DRAM*)Ж память в виде массивов конденсаторов.

Память динамического типа

Динамическая оперативная память (DRAM – Dynamic Random Access Memory) – энергозависимая полупроводниковая память с произвольным доступом.

На данный момент – это основной тип оперативной памяти, используемый в современных персональных компьютерах и обеспечивающий наилучший показатель отношения цена-качество по сравнению с другими типами оперативной памяти. Изготавливается в виде модулей памяти, которые устанавливаются на системной плате компьютера.

Память динамического типа (продолжение)

Сейчас ОЗУ выполняется по технологии **DRAM** в форм-факторах **DIMM и SO-DIMM**.

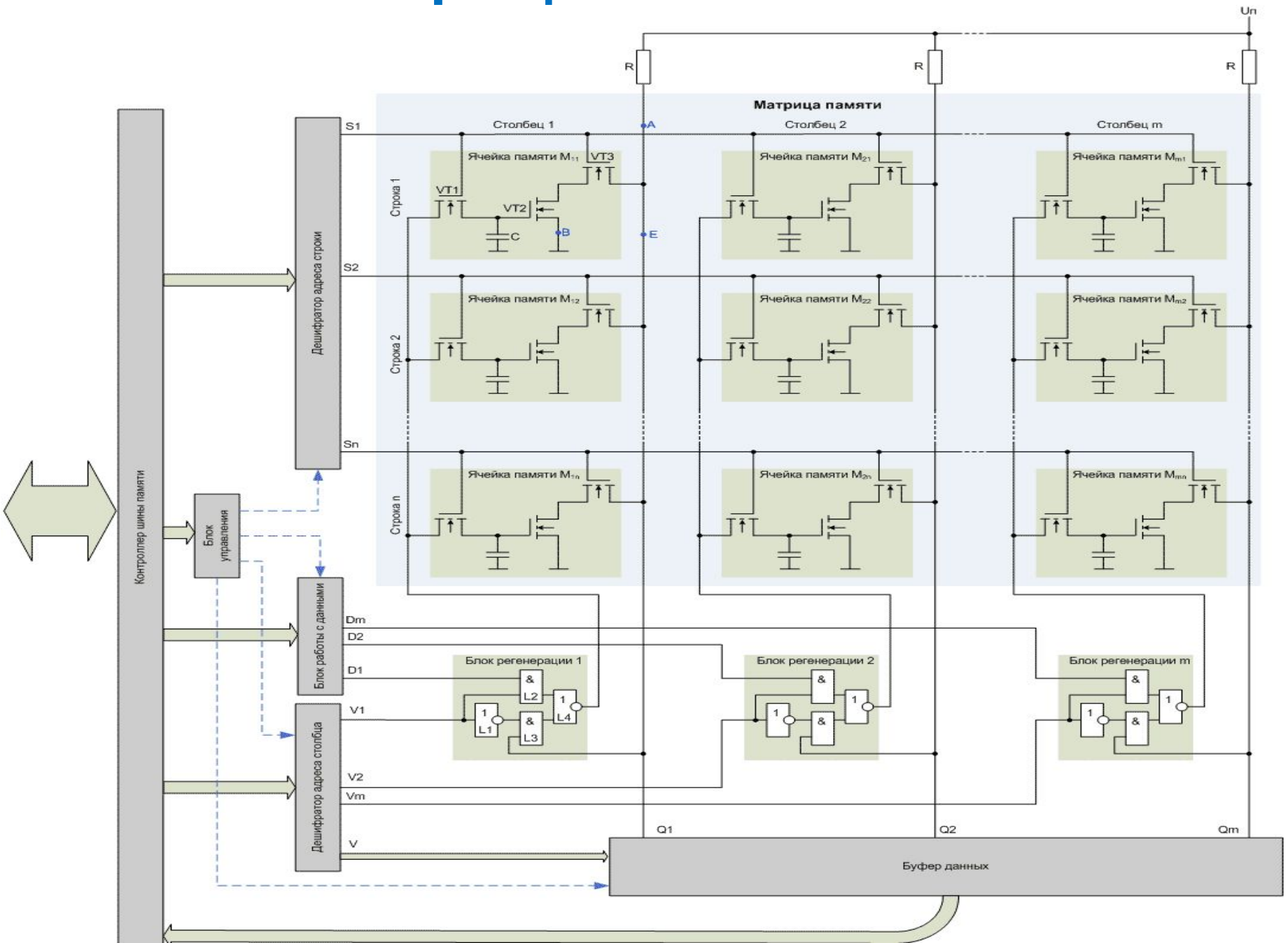
Память форм-фактора SO-DIMM предназначена для использования в ноутбуках, компактных ITX-системах, моноблоках - словом там, где важен минимальный физический размер модулей памяти. Отличается от форм-фактора DIMM уменьшенной примерно в 2 раза длиной модуля, и меньшим количеством контактов на плате (204 и 360 контактов у SO-DIMM DDR3 и DDR4 против 240 и 288 на платах тех же типов DIMM-памяти).

DIMM - оперативная память для полноразмерных компьютеров.

Устройство DRAM

Каждая ячейка **DRAM** которой состоит из одного конденсатора и нескольких транзисторов. Конденсатор хранит один бит данных, а транзисторы играют роль ключей, удерживающих заряд в конденсаторе и разрешающих доступ к конденсатору при чтении и записи данных.
Однако транзисторы и конденсатор – неидеальные, и на практике заряд с конденсатора достаточно быстро истекает. Поэтому периодически, несколько десятков раз в секунду, приходится дозаряжать конденсатор. К тому же процесс чтения данных из динамической памяти – деструктивен, то есть при чтении конденсатор разряжается, и необходимо его заново подзаряжать, чтобы не потерять навсегда данные, хранящиеся в ячейке памяти.

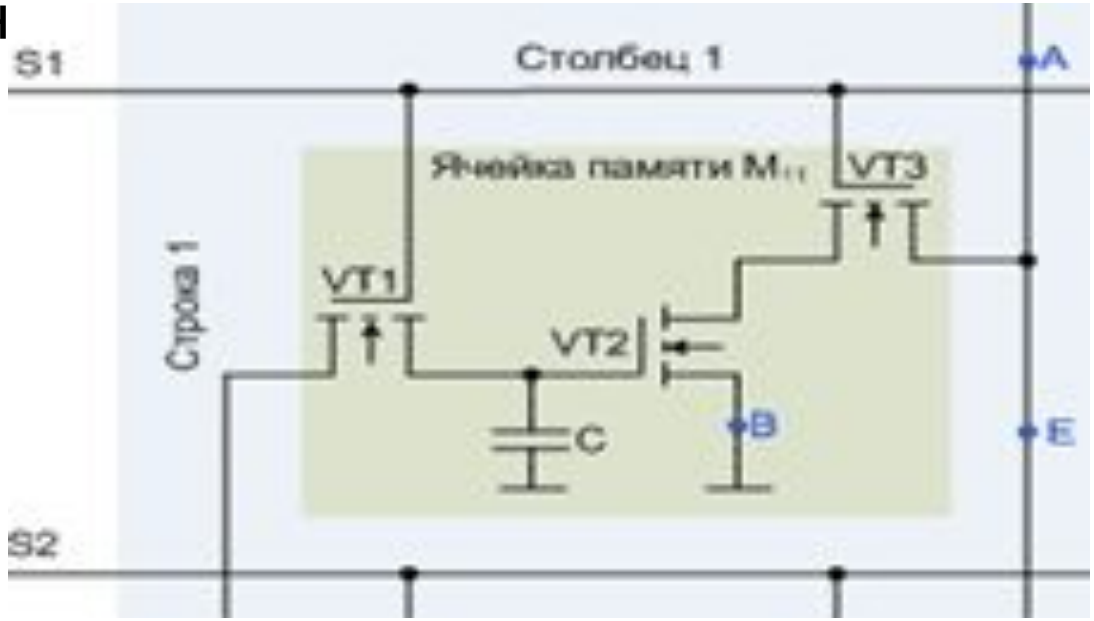
Упрощенная схема



Описание схемы

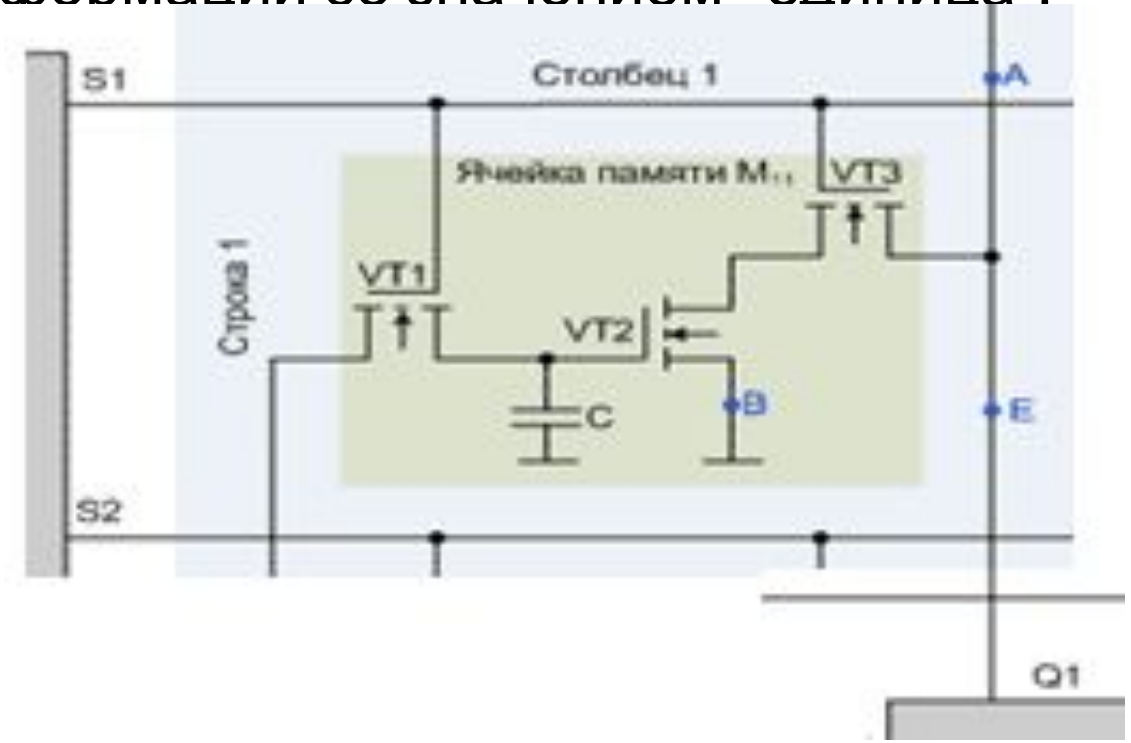
Основным блоком памяти является матрица памяти, состоящая из множества ячеек, каждая из которых хранит 1 бит информации.

Каждая ячейка состоит из одного конденсатора (С) и трех транзисторов. Транзистор VT1 разрешает или запрещает запись новых данных или регенерацию ячейки. Транзистор VT3 выполняет роль ключа, удерживающего конденсатор от разряда и разрешающего или запрещающего чтение данных из ячейки памяти. Транзистор VT2 используется для считывания данн



Описание схемы

Если на конденсаторе есть заряд, то транзистор VT2 открыт, и ток пойдет по линии AB, соответственно, на выходе Q1 тока не будет, что означает – ячейка хранит бит информации с нулевым значением. Если заряда на конденсаторе нет, то конденсатор VT2 закрыт, а ток пойдет по линии AE, соответственно, на выходе Q1 ток будет, что означает – ячейка хранит бит информации со значением “единица”.



Описание схемы (продолжение)

Заряд в конденсаторе, используемый для поддержания транзистора VT2 в открытом состоянии, во время прохождения по нему тока, быстро расходуется, поэтому при чтении данных из ячейки необходимо проводить регенерацию заряда конденсатора.

Для работы динамической памяти на матрицу должно всегда поступать напряжение, на схеме оно обозначено, как Uп. С помощью резисторов R напряжение питания Uп равномерно распределяется между всеми столбцами матрицы. Также в состав памяти входит контроллер шины памяти, который получает команды, адрес и данные от внешних устройств и ретранслирует их во внутренние блоки памяти. Команды передаются в блок управления, который организует работу остальных блоков и периодическую регенерацию ячеек памяти

Описание схемы (продолжение)

Адрес преобразуется в две составляющие – адрес строки и адрес столбца, и передается в соответствующие дешифраторы. Дешифратор адреса строки определяет, с какой строки надо провести чтение или запись, и выдает на эту строку напряжение. Дешифратор адреса столбца при чтении данных определяет, какие из считанных бит данных были запрошены и должны быть выданы в шину памяти. При записи данных дешифратор определяет, в какие столбцы надо подать команды записи.

Описание схемы (продолжение)

Блок работы с данными определяет, какие данные, в какую ячейку памяти требуется записать, и выдает соответствующие биты данных для записи в эти ячейки. Блоки регенерации определяют: когда происходит чтение данных и надо провести регенерацию ячейки, из которой данные были считаны; когда происходит запись данных, а, следовательно, регенерацию ячейки производить не надо. Буфер данных сохраняет всю считанную строку матрицы, так как при чтении всегда считывается вся строка целиком, и позволяет потом выбрать из считанной строки требуемые биты данных.

Особенности DRAM

DRAM дешевле SRAM (один конденсатор и один транзистор на 1 бит дешевле нескольких транзисторов триггера) и занимает меньшую площадь на кристалле (там, где в SRAM размещается один триггер, хранящий 1 бит, можно разместить несколько конденсаторов и транзисторов для хранения нескольких бит). Но *DRAM* имеет и недостатки. Работает медленнее, поскольку, если в *SRAM* изменение управляющего напряжения на входе триггера очень быстро изменяет его состояние, то для того, чтобы изменить состояние конденсатора, его нужно зарядить или разрядить. Перезаряд конденсатора гораздо более длителен (в 10 и более раз), чем переключение триггера, даже если ёмкость конденсатора очень мала. Второй существенный недостаток — конденсаторы со временем разряжаются. Причём разряжаются они тем быстрее, чем

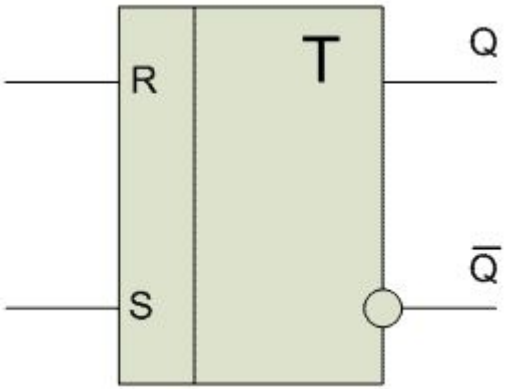
Особенности DRAM (продолжение)

Именно из-за того, что заряд конденсатора динамически уменьшается во времени, память на конденсаторах получила своё название *DRAM* — динамическая память. Поэтому, чтобы не потерять содержимое памяти, заряд конденсаторов периодически восстанавливается («регенерируется») через определённое время, называемое циклом регенерации (обычно 2 мс). Для регенерации в современных микросхемах достаточно выполнить циклограмму «чтения» по всем строкам запоминающей матрицы. Процедуру регенерации выполняет процессор или контроллер памяти. Так как для регенерации памяти периодически приостанавливается обращение к памяти, это снижает среднюю скорость обмена с этим видом ОЗУ.

Статическая оперативная память

Статическая память (SRAM) – это энергозависимая полупроводниковая память с произвольным доступом, в которой каждый разряд хранится в триггере, позволяющем поддерживать состояние разряда без постоянной перезаписи. Для организации чтения и записи из ячейки памяти дополнительно используется три или более транзисторов.

Устройство триггера



Триггер – это элемент памяти с двумя стабильными состояниями – «0» и «1». В установленном состоянии триггер сохраняется, пока на него подается питание.

Обычно триггер имеет два входа:

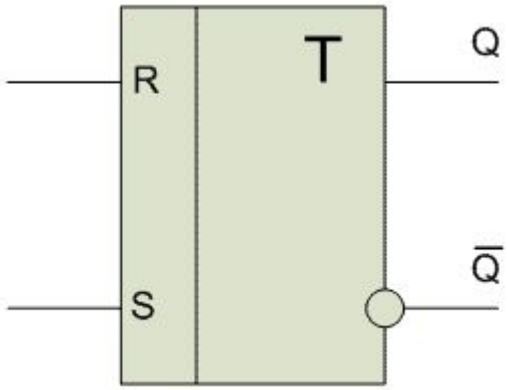
R (Reset) – сбросить триггер (установить в состояние «0»), S (Set) – установить триггер в состояние «1», $\overline{Q}$

и два выхода: Q и инвертированное Q ( ).

Входы R и S используются для установки состояния триггера. Если на вход S подать напряжение, соответствующее логической единице (далее просто логическую единицу), а на вход R – напряжение, соответствующее логическому нулю (далее просто логический ноль), то триггер перейдет в состояние единицы и сохранит это состояние даже, если на вход S перестать подавать

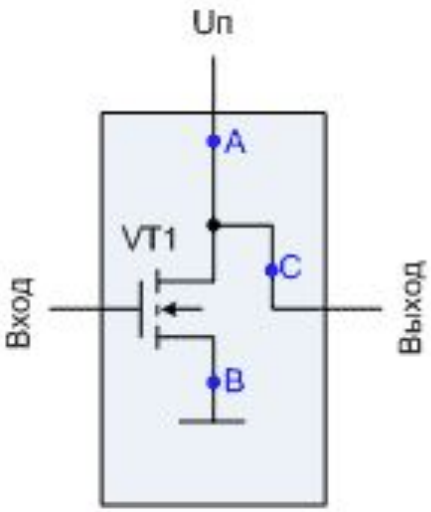
Устройство триггера (продолжение)

Если на вход S подать логический ноль, а на вход R – логическую единицу, то триггер перейдет в состоянии сохранения нуля.
При подаче на оба входа логического нуля, состояние триггера не измениться.
При подаче на оба входа логической единицы, в общем случае состояние триггера будет неопределенно, то есть неизвестно, в какое состояние он перейдет.
На выходах Q $\bar{Q}$ можно прочитать установленное состояние триггера.



Структурная схема инвертора

Триггер состоит из двух инвертеров (логических элементов «НЕ»), причем выход одного инвертера замкнут на вход другого.



На рисунке представлена простейшая схема реализации инвертера, состоящая из одного транзистора. Рассмотрим, как он работает.

На элемент всегда подается питание Uп. В результате, создаваемый ток может пойти либо по линии АВ, в этом случае на выходе инвертера ток будет отсутствовать (будет логический ноль), либо – по линии АС, в этом случае на выходе инвертера ток будет присутствовать (будет логическая единица).

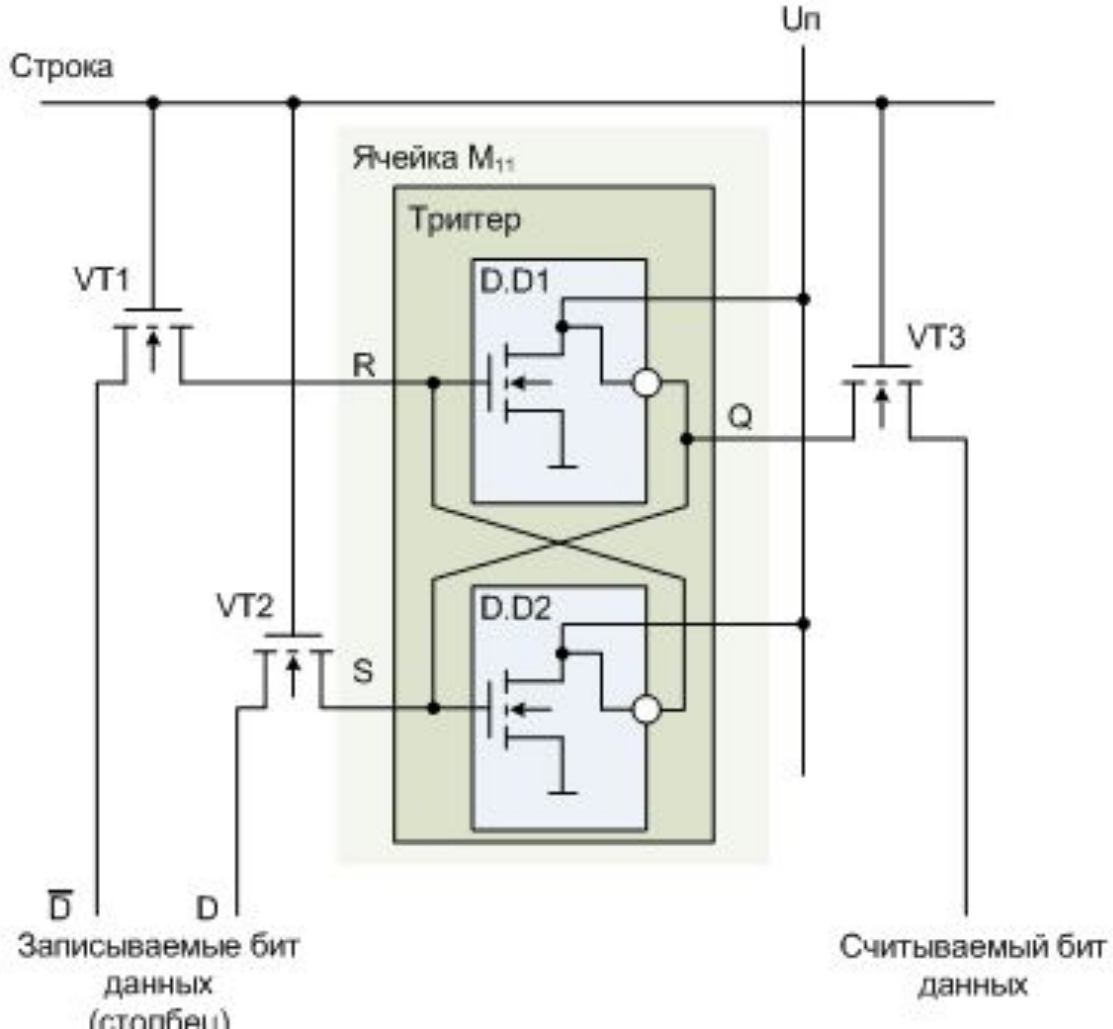
По линии АВ ток пойдет, если транзистор VT1 будет открыт, а для этого необходимо подать напряжение на вход инвертера.

По линии АС ток пойдет, если транзистор VT1 будет закрыт, а это произойдет при отсутствии напряжении на входе инвертера.

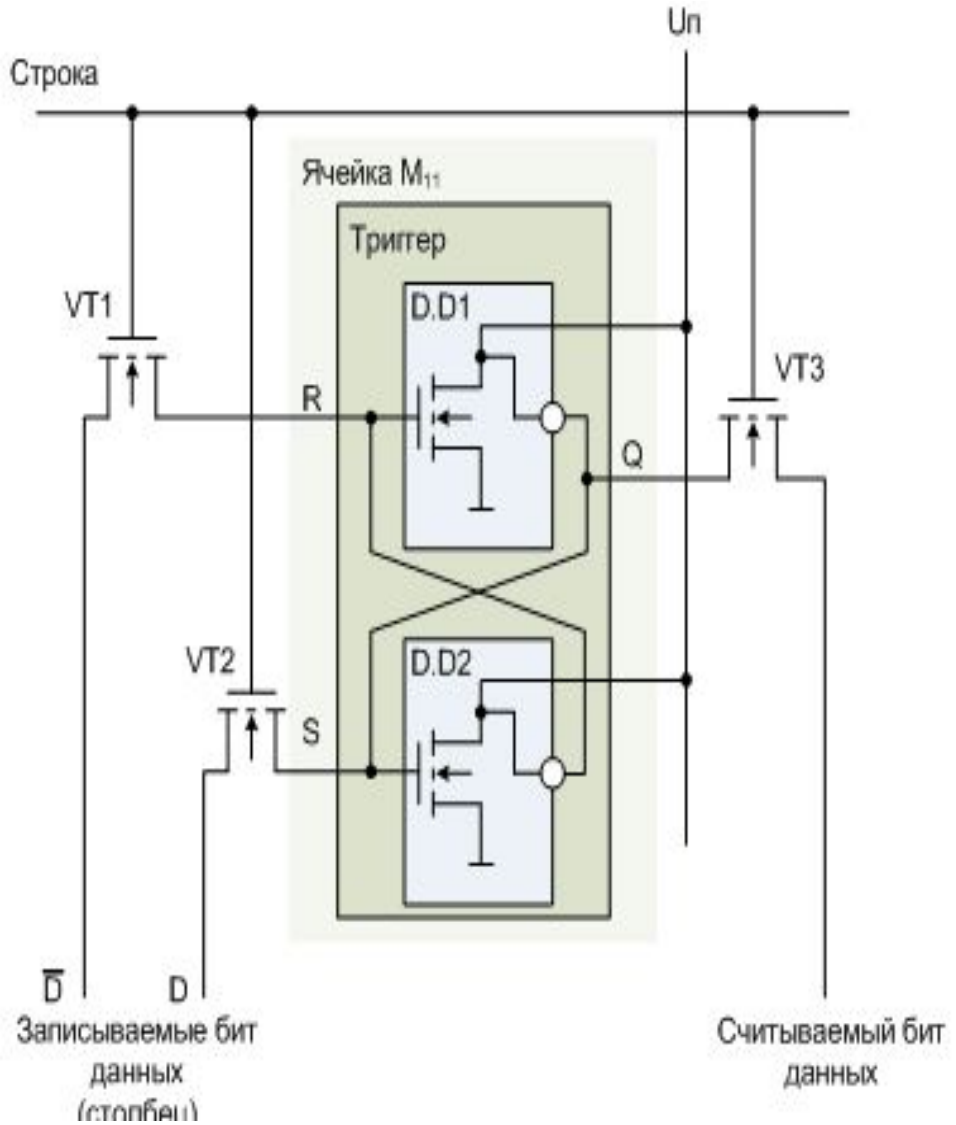
Таким образом, если на вход инвертера подается логическая единица, то на выходе будет логический ноль. И, соответственно, при подаче на вход инвертера логического нуля, на выходе будет поучена логическая единица.

Устройство ячейки статической памяти

На рисунке приведена упрощенная схема одного из способов организации ячейки статической памяти. Она состоит из одного триггера и трех транзисторов, выполняющих роль ключей, открывающих и закрывающих доступ к ячейке памяти. Транзисторы VT1 и VT2 используются для разрешения и запрета записи в ячейку, а транзистор VT3 – для разрешения и запрета чтения.



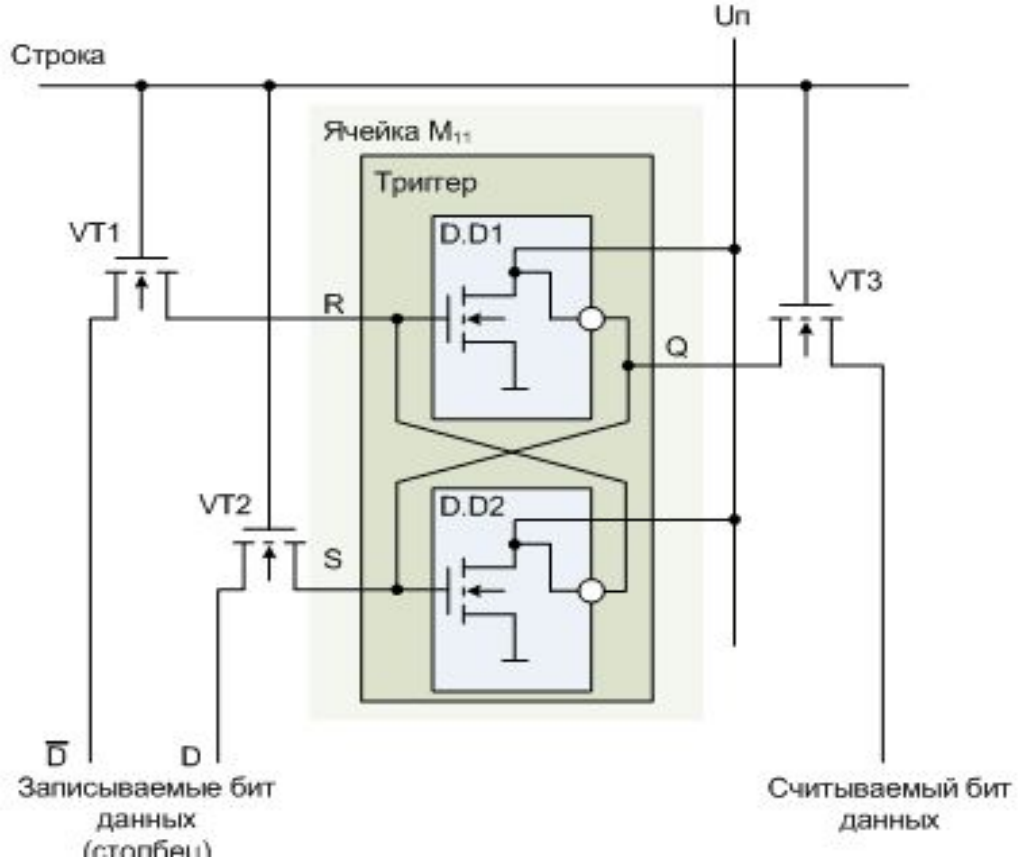
Устройство ячейки статической памяти



Для записи данных необходимо подать напряжение в линию строки, после чего транзисторы VT1, VT2 и VT3 откроются. Затем для записи единицы необходимо подать напряжение, соответствующее логической единице, на линию D и напряжение, соответствующее логическому нулю, на линию $\overline{D}$.
Для переключения триггера в состояние хранения нуля необходимо подать напряжение, соответствующее логическому нулю, на линию D и напряжение, соответствующее логической единице, на линию $\overline{D}$.
В установленном состоянии триггер будет оставаться и после снятия напряжения с линии строки и с линий D и до тех пор, пока на него будет подаваться питание Un.

Устройство ячейки статической памяти (продолжение)

Для считывания данных необходимо на выходы D и подать напряжение, соответствующее логическому нулю, так как подача двух логических нулей на входы триггера не изменит его состояния, а затем подать напряжение на строку. В результате, транзистор VT3 откроется, и ток с триггера по линии Q пройдет в устройство считывания. Одновременно с транзистором VT3 откроются транзисторы VT1 и VT2. Но так как напряжение на линиях D и соответствует логическому нулю, то оно не повлияет на состояние транзистора.



Считывание данных с ячейки статической памяти, в отличие от чтения с ячейки динамической памяти, не приводит к потере сохраненного бита данных, поэтому, перезапись данных в ячейку статической памяти не требуется.

Достоинства и недостатки статической памяти

Достоинства:

- высокая скорость работы;
- нет необходимости регенерации ячеек.

Недостатки:

- высокая цена;
- низкая плотность упаковки;
- небольшой объем;
- высокое энергопотребление.

В связи с перечисленными выше достоинствами и недостатками, область применения статической памяти ограничивается, в основном, использованием ее в качестве КЭШ-памяти, что позволяет при небольшом увеличении стоимости уменьшить влияние недостатков динамической памяти на производительность ЭВМ.

Основные характеристики оперативной памяти

Основными характеристиками оперативной памяти являются:

- ***Тактовая частота работы (Frequency)***
- ***Объем (Capacity)***
- ***Тип памяти (Memory Type)***
- ***Рабочее напряжение (Current Voltage)***
- ***Задержки (Timing's)***
- ***Фирма производитель (Brand)***

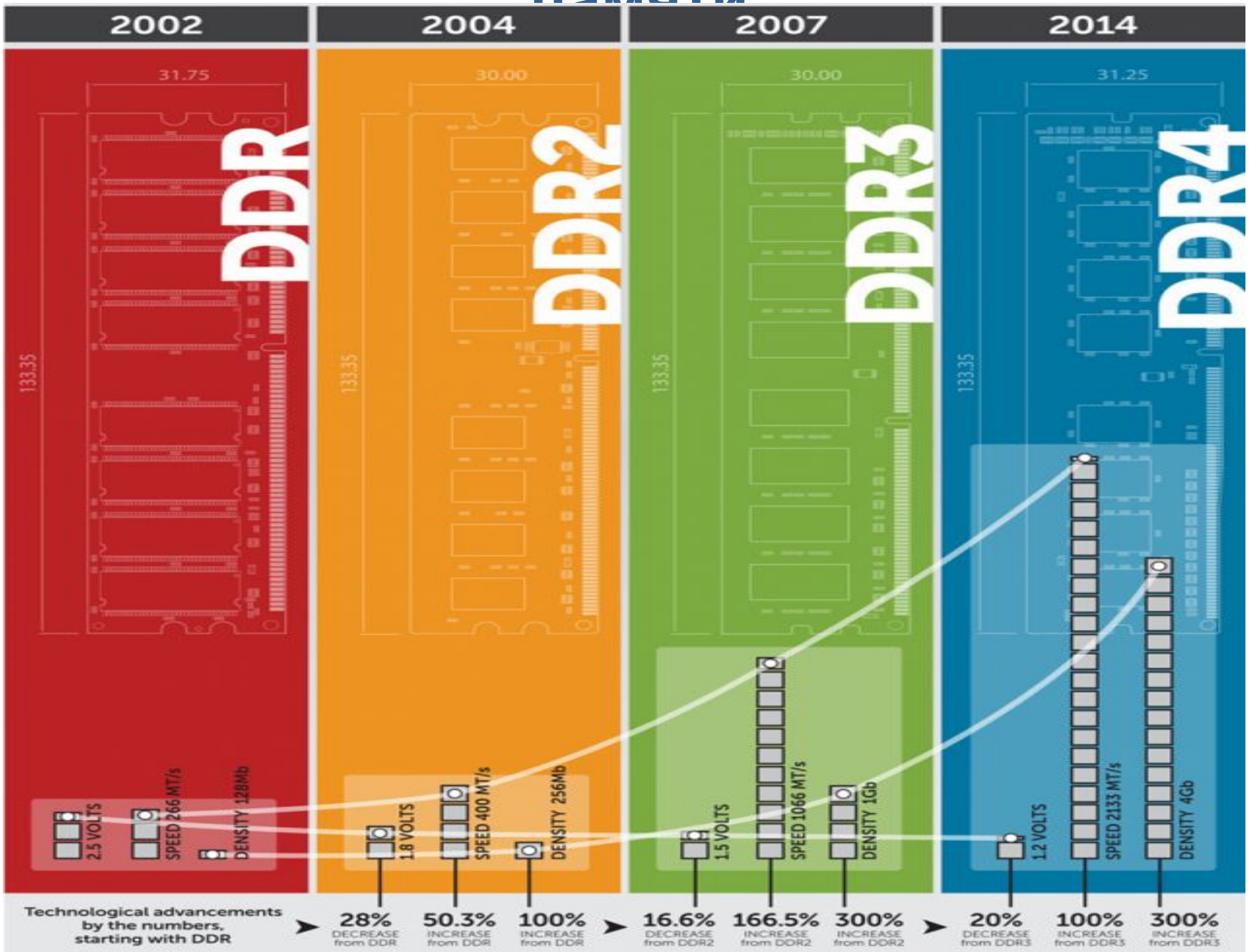
1. Тактовая частота (Frequency) – данный параметр указывает рабочую частоту модуля памяти, т.е. это частота обмена данными модуля памяти с CPU. Единицей измерения данного параметра, является МГц (MHz). Это скорость обмена модуля памяти с центральным процессором.

2. Объем (Capacity) – параметр, указывающий физический объем модуля, т.е. это адресное пространство, для хранения данных. Единица измерения Гб (Gb)

Основные характеристики оперативной памяти

3. Тип памяти (Type) – DDR, DDR2, DDR3, DDR4
Каждый тип памяти, должен быть совместим с типом, который поддерживается материнской платой.
4. Рабочее напряжение (Current Voltage) – параметр, показывающий номинальное напряжение на модуле оперативной памяти.
5. Задержки (Timing's) – представляют собой временные интервалы, необходимые для записи, перезаписи, обнуления и т.д. памяти. При выборе памяти следует искать модули памяти, которые имеют меньшие задержки. Однако имеет место быть следующая ситуация – модуль памяти имеющий высокие рабочие частоты, обычно имеет задержки выше чем низкочастотные.
6. Фирма производитель (Brand).

Основные характеристики оперативной памяти



Сравнение DDR3 и DDR4

Описание	DDR3	DDR4	Advantage
Плотность размещения микросхем	512 - 8 Гбит	4 - 16 Гбит	Повышенная емкость DIMM
Скорость передачи данных	800 - 2133 Мбит/с	1600 – 3200 Мбит/с	Переход к повышенной скорости ввода-вывода
Напряжение	1,5 В	1,2 В	Пониженное энергопотребление памяти
Стандарт низкого напряжения	Да (DDR3L при 1,35 В)	1,1 В	Снижение мощности памяти
Внутренние банки	8	16	Другие банки
Группы банков (BG)	0	4	Более быстрый произвольный доступ
Задержка чтения	AL + CL	AL + CL	Увеличенные значения
Задержка записи	AL + CWL	AL + CWL	Увеличенные значения
Контакты DIMM	240 (R, LR, U); 204 (SODIMM)	288 (R, LR, U); 260 (SODIMM)	
RAS	ECC	CRC, четность, адресуемость, GDM	Дополнительные функции RAS; увеличенная целостность данных

Регистровая память

- **Регистры** - специальные ячейки памяти, расположенные физически внутри процессора, доступ к которым осуществляется не по адресам, как к основной памяти, а по именам.

Регистры архитектуры x86

- ***Пользовательские регистры***

Пользовательские регистры называются так потому, что программист может использовать их при написании своих программ.

Регистры архитектуры x86

К пользовательским регистрам относятся:

- восемь 32-битных регистров, которые могут использоваться программистами для хранения данных и адресов (их еще называют *регистрами общего назначения* (РОН)):
eax/ax/ah/al; **ebx/bx/bh/bl**; **edx/dx/dh/dl**; **ecx/cx/ch/cl**; **ebp/bp**; **esi/si**; **edi/di**; **esp/sp**.
Все регистры этой группы позволяют обращаться к своим младшим частям. Для самостоятельной адресации можно использовать только младшие 16 и 8-битные части этих регистров. Старшие 16 бит этих регистров как самостоятельные объекты недоступны. Это сделано, для совместимости с младшими 16-разрядными моделями микропроцессоров фирмы Intel.

Пользовательские регистры

- шесть регистров сегментов: **cs, ds, ss, es, fs, gs**;
- регистры состояния и управления:
  регистр флагов **eflags/flags**;
  регистр указателя команды **eip/ip**.
- регистры сопроцессора x87 и расширения MMX
- регистры расширения XMM

Сегментные регистры

В программной модели микропроцессора имеется шесть сегментных регистров: *cs, ss, ds, es, gs, fs*. Их существование обусловлено спецификой организации и использования оперативной памяти микропроцессорами Intel. Она заключается в том, что микропроцессор аппаратно поддерживает структурную организацию программы в виде трех частей, называемых *сегментами*. Соответственно, такая организация памяти называется ***сегментной***.

Для того чтобы указать на сегменты, к которым программа имеет доступ в конкретный момент времени, и предназначены *сегментные регистры*. Фактически, с небольшой поправкой, в этих регистрах содержатся адреса памяти, с которых начинаются соответствующие сегменты.

Сегментные регистры

Логика обработки машинной команды построена так, что при выборке команды, доступе к данным программы или к стеку неявно используются адреса во вполне определенных сегментных регистрах. Микропроцессор поддерживает следующие типы сегментов:

Сегмент кода. Содержит команды программы. Для доступа к этому сегменту служит регистр **cs** (code segment register) - *сегментный регистр кода*. Он содержит адрес сегмента с машинными командами, к которому имеет доступ процессор (то есть эти команды загружаются в конвейер микропроцессора).

Сегментные регистры

Сегмент данных. Содержит обрабатываемые программой данные.
Для доступа к этому сегменту служит регистр **ds** (data segment register) - *сегментный регистр данных,* который хранит адрес сегмента данных текущей программы.
Сегмент стека. Этот сегмент представляет собой область памяти, называемую *стеком*.
Работу со стеком микропроцессор организует по следующему принципу: *последний записанный в эту область элемент выбирается первым*. Для доступа к этому сегменту служит регистр **ss** (stack segment register) - *сегментный регистр стека,* содержащий адрес сегмента стека.

Сегментные регистры

Дополнительный сегмент данных.
Большинство машинных команд предполагают, что обрабатываемые ими данные расположены в сегменте данных, адрес которого находится в регистре *ds*.
Если программе недостаточно одного сегмента данных, то она имеет возможность использовать еще три дополнительных сегмента данных. Но в отличие от основного сегмента данных, адрес которого содержится в сегментном регистре *ds*, при использовании дополнительных сегментов данных их адреса требуется указывать явно с помощью специальных *префиксов переопределения сегментов* в команде. Адреса дополнительных сегментов данных должны содержаться в регистрах **es, gs, fs** (extension data segment registers).

Регистры состояния и управления

В микропроцессор включены несколько регистров, которые постоянно содержат информацию о состоянии как самого микропроцессора, так и программы, команды которой в данный момент загружены на конвейер. К этим регистрам относятся:

- **регистр флагов** **eflags/flags**;
- **регистр указателя команды** **eip/ip**.

Используя эти регистры, можно получать информацию о результатах выполнения команд и влиять на состояние самого микропроцессора. Разрядность *eflags/flags* - 32/16 бит. Отдельные биты данного регистра имеют определенное функциональное назначение и называются флагами.

Регистры состояния и управления

Регистр *eip/ip* имеет разрядность 32/16 бит и содержит смещение следующей подлежащей выполнению команды относительно содержимого сегментного регистра cs в текущем сегменте команд. Этот регистр непосредственно недоступен программисту, но загрузка и изменение его значения производятся различными командами управления, к которым относятся команды условных и безусловных переходов, вызова процедур и возврата из процедур. Возникновение прерываний также приводит к модификации регистра *eip/ip*.

Регистры сопроцессора x87

- Сопроцессор (FPU) предназначен для выполнения операций над вещественными числами. С программной точки зрения сопроцессор содержит блок регистров данных, регистр управления и группу регистров состояния и указателей. Восемь регистров данных разрядностью 80 бит организованы в стек. Номер регистра, являющегося текущей вершиной стека, хранится в специальном поле регистра состояния (указателе вершины стека). Операция push уменьшает значение указателя на 1 и помещает в стек данные в регистр, являющийся новой вершиной стека.

Регистры сопроцессора x87

- Операция pop записывает данные с вершины стека в память или регистр и увеличивает указатель на 1. Инструкции адресуют регистры либо явно, либо неявно. Неявная адресация подразумевает операнд, находящийся на вершине стека. Явная адресация подразумевает указание смещения регистра относительно вершины стека - st(i).

Расширение MMX

- MMX было первым расширением, реализующим технологию SIMD (Single Instruction - Multiple Data). Основная идея SIMD заключается в одновременной обработке нескольких элементов данных одной операцией. Расширение MMX использует новые типы упакованных **64-битные** целочисленных данных. Эти типы данных могут специальным образом обрабатываться в 64-битных регистрах MM0-MM7, представляющих собой младшие биты стека 80-битных регистров FPU. Каждая инструкция MMX выполняет действие сразу над всем комплектом операндов (8, 4, 2 или 1), размещенных в адресуемых регистрах. Как и регистры FPU, эти регистры не могут использоваться для адресации памяти. Совпадение регистров MMX и FPU накладывает ограничение на чередование кодов FPU и MMX.

Блок XMM

Начиная с Pentium III, Intel использует в своих процессорах новое потоковое расширение SSE (Streaming SIMD Extension). Оно реализуется дополнительным независимым блоком, имеющим восемь **128-битных** регистров, названных XMM0-XMM7, и регистр состояния/управления MXCSR. В каждый из регистров XMM помещаются четыре числа в формате с плавающей точкой одинарной точности. Блок позволяет выполнять векторные (пакетные) и скалярные инструкции. Векторные инструкции реализуют операции сразу над четырьмя комплектами операндов. Скалярные инструкции работают только с одним комплектом операндов - младшим 32-битным словом. При выполнении инструкций XMM традиционное оборудование FPU/MMX не используется, что позволяет эффективно смешивать инструкции MMX с инструкциями с плавающей точкой

Регистры архитектуры x86-64 (AMD64)

Выпущенные фирмой AMD процессоры Athlon64 и Opteron имеют архитектуру x86-64, которая в отличие от архитектуры x86 является полностью 64-битной. Она естественным образом расширяет регистры общего назначения x86 до 64 битов и увеличивает их количество. Также удваивается число регистров XMM. Регистры FPU/MMX остаются без изменений.

В отличие от процессоров x86, у которых все вещественные вычисления производились в сопроцессоре, а блоку XMM отводились только векторные операции, процессоры x86-64 практически все вещественные вычисления выполняют в блоке XMM.

Кэширование памяти

Кэшем или кэш-памятью называют специальное хранилище часто используемых данных, доступ к которому осуществляется в десятки, сотни и тысячи раз быстрее, чем к оперативной памяти или другому носителю информации. Все приложения, вэб-браузеры, аудио- и видеоплееры, редакторы баз данных, компоненты операционной системы и оборудования, а именно cache L1-L3 центрального процессора, фреймбуфер графического чипа, буферы накопителей и прочие, имеют собственную кэш память. Но ее реализация у вышеперечисленных "элементов" будет разной: аппаратной или программной.

Кэширование памяти

Кеш программ – это отдельная папка или файл, куда загружаются, например, картинки, меню, скрипты, мультимедийный контент и прочее содержимое посещенных сайтов. Именно в такую папку в первую очередь обращается браузер, когда открываете веб-страницу повторно. Подкачка части контента из локального хранилища ускоряет ее загрузку и уменьшает сетевой трафик (кэш браузера). В накопителях в том числе и в жестких дисках кэш представляет собой отдельную микросхему RAM емкостью 1-256Mb, который располагается на плате электроники. В него поступает информация, считанная с магнитного слоя и пока не загруженная в оперативную память, а также данные, которые чаще всего запрашивает операционная система (кэш диска).

Кэширование памяти

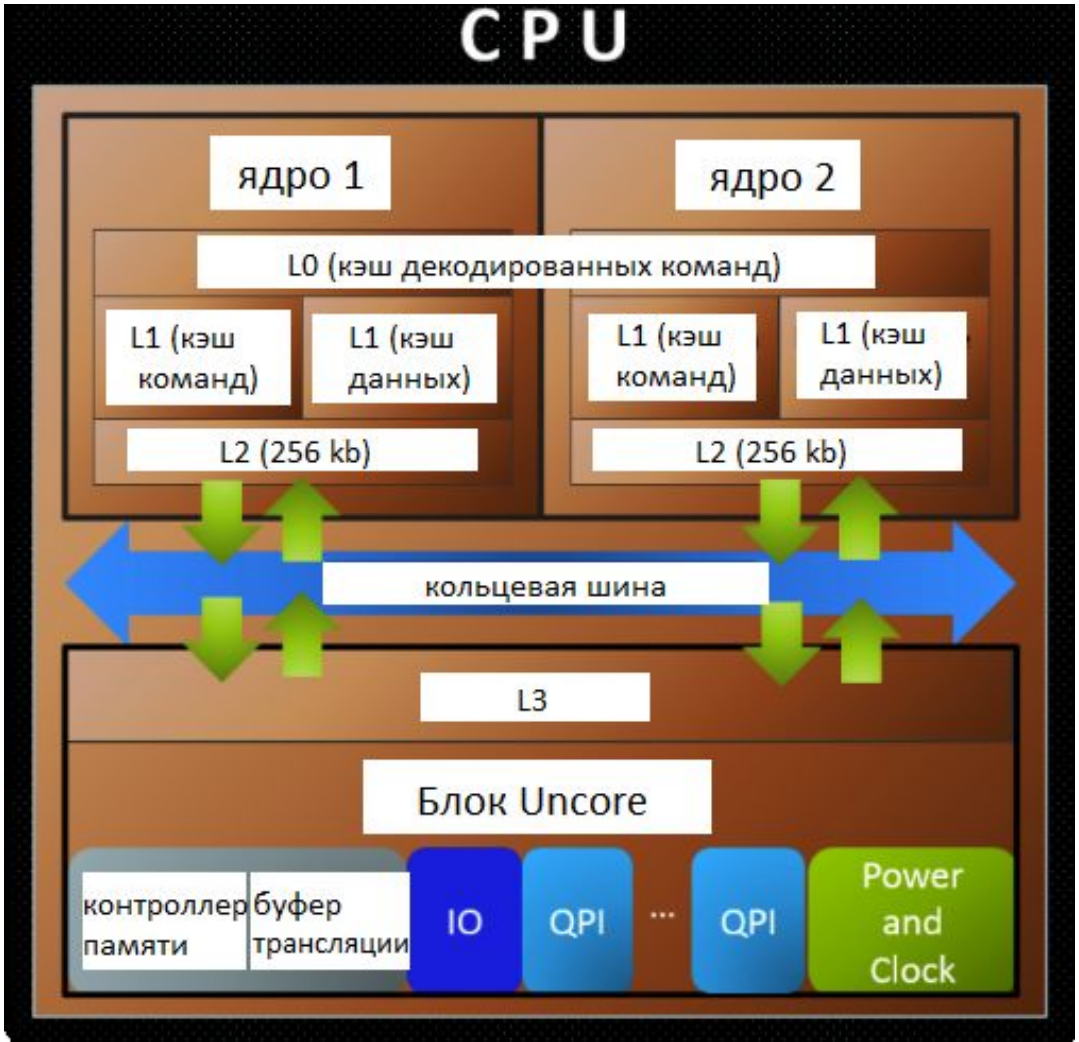
Современный центральный процессор содержит 2-3 основных уровня кэш-памяти которая еще называется сверхоперативной памятью. Размещены они в виде аппаратных модулей на одном с ним кристалле. Самым быстрым и наименьшим по объему (32-64Kb) является cache Level 1 (L1) – он работает на той же частоте, что и процессор. L2 занимает среднее положение по скорости и емкости (от 128 Kb до 12 Mb). А L3 – самый медленный и объемный (до 40 Mb), на некоторых моделях отсутствует. Скорость L3 является низкой лишь относительно его более быстрых собратьев, но и он в сотни раз быстрее самого производительного ОЗУ. В сверхоперативной памяти процессора хранятся данные, которые используются постоянно. Они перекачены из ОЗУ и инструкций машинного кода. Чем больше такой памяти, тем процессор работает быстрее.

Кэширование памяти

На сегодняшний день, три уровня кэширования это не предел. Корпорация Intel использует архитектуру Sandy Bridge. Благодаря ей, стал доступен дополнительный кэш "cache L0”. Данный раздел отвечает за хранение расшифрованных микрокоманд. А наиболее высокопроизводительные ЦП (серверы) имеют и кэш четвертого уровня, выполненный в виде отдельной микросхемы.

Кэширование памяти

Схематично взаимодействие уровней cache L0-L3 выглядит так (на примере Intel Xeon): кэш процессора 4 уровня



Кэширование памяти

Оперативная память персонального компьютера реализуется на относительно медленной динамической памяти DRAM. Обращение к DRAM могло бы приводить к простою процессора и появлению тактов ожидания. Статическая память, построенная на триггерных ячейках, имеет с малое время доступа, но и высокую стоимость. Поэтому пришли к компромиссу - это сочетание основной оперативной памяти большого объема на DRAM со временем доступа от 60 - 100 нс, с относительно небольшой кэш памятью на быстродействующих микросхемах SRAM со временем доступа от 5 - 20 нс.

Кэш память

Кэш является дополнительным быстродействующим хранилищем копий блоков информации из основной области. Поскольку кэш не может хранить копию всей основной памяти, т.к. его объем во много раз меньше основной памяти, то в нем хранятся блоки памяти вероятность обращения к которым, в ближайшее время, достаточно велика. Внутренняя кэш-память, включена в сам процессор для повышения производительности системы.
Кэширование применяется в процессорах, жестких дисках, браузерах, веб-серверах и т.п.

Кэш память

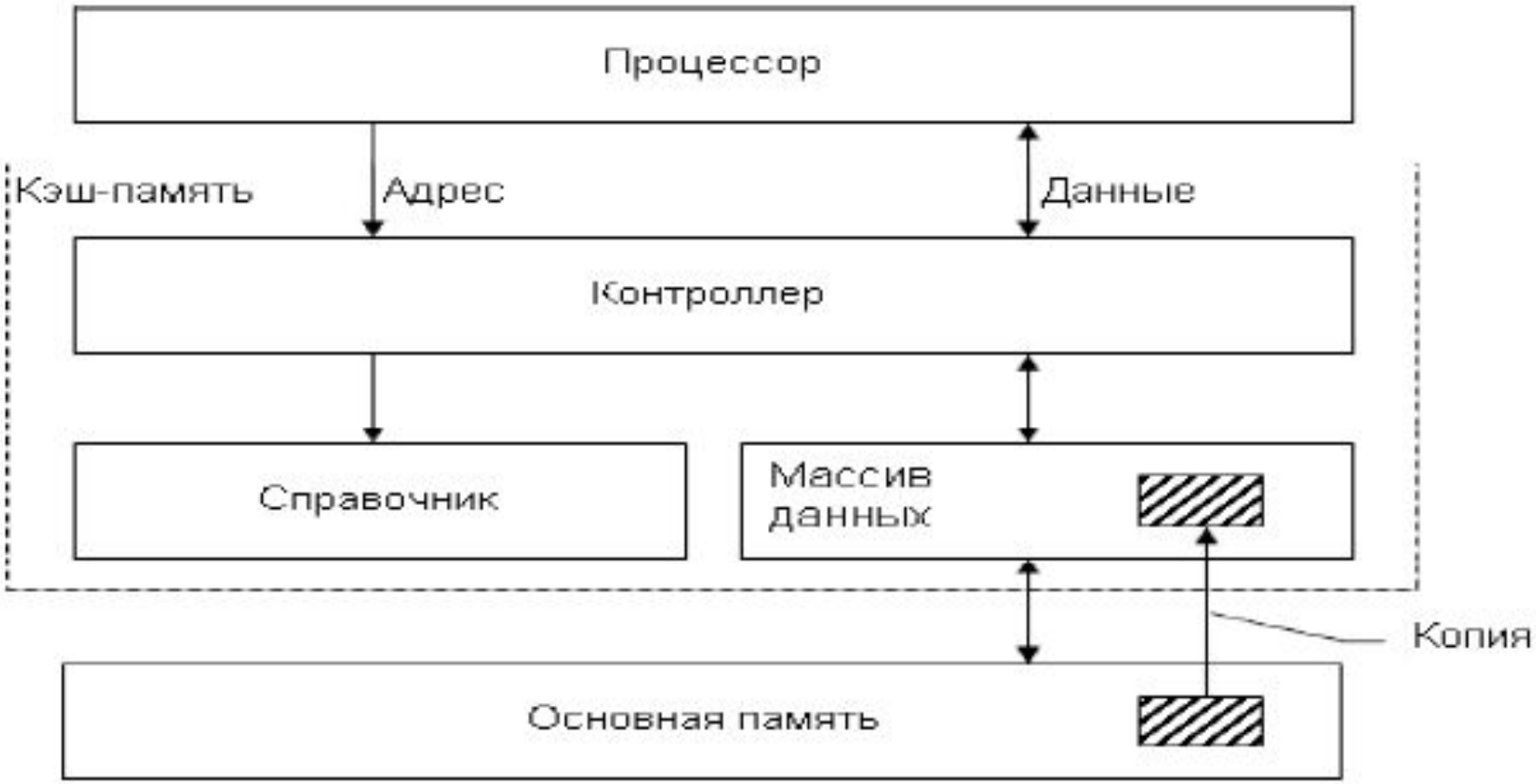
Кэш, используемый микропроцессором компьютера для уменьшения среднего времени доступа к компьютерной памяти, использует небольшую, очень быструю память (обычно типа SRAM), которая хранит копии часто используемых данных из основной памяти. Если большая часть запросов в память будет обрабатываться кэшем, средняя задержка обращения к памяти будет приближаться к задержкам работы кэша.
Когда процессору нужно обратиться в память для чтения или записи данных, он сначала проверяет, доступна ли их копия в кэше. В случае успеха проверки процессор производит операцию, используя кэш, что значительно быстрее использования более медленной основной оперативной памяти.

Кэш память

Данные между кэшем и памятью передаются блоками фиксированного размера, также называемые **линиями кэша** (*cache line*) или блоками кэша.
Большинство современных микропроцессоров для компьютеров и серверов имеют как минимум три независимых кэша: **кэш инструкций** для ускорения загрузки машинного кода, **кэш данных** для ускорения чтения и записи данных и буфер ассоциативной трансляции (TLB) для ускорения трансляции виртуальных (логических) адресов в физические, как для инструкций, так и для данных.
Увеличение размера кэш-памяти может положительно влиять на производительность почти всех приложений, хотя в некоторых случаях эффект незначителен.

Кэш память

Кэш-память состоит из : массива данных, справочника или каталога (cache directory) и контроллера (устройства управления).



Кэш память

В массив данных копируются блоки основной памяти, а их адреса заносятся в каталог. Каталог содержит список текущего соответствия блоков данных областям основной памяти. При каждом обращении к памяти контроллер кэш-памяти по каталогу проверяет, есть ли действительная копия затребованных данных в кэше. Если она там есть, то реализуется *кэш-попадание* (cache hit), и данные берутся из кэш-памяти. Если действительной копии там нет, то реализуется *кэш-промах* (cache miss), и данные берутся из основной памяти и помещаются в кэш-память.

Кэш память

ОП состоит из 2n адресуемых слов, где каждое слово имеет уникальный n-разрядный адрес. При взаимодействии с кэшем эта память рассматривается как M блоков фиксированной длины по k слов в каждом (M = 2n/k). Кэш-память состоит из m блоков аналогичного размера (блоки в кэш-памяти принято называть *строками*), причем их число значительно меньше числа блоков в основной памяти (m << M). При считывании слова из какого-либо блока ОП этот блок копируется в одну из строк кэша. Поскольку число блоков ОП больше числа строк, отдельная строка не может быть выделена постоянно одному и тому же блоку ОП.

Кэш память

С каждой строкой кэша связана информация об адресе скопированного в нее блока основной памяти и ее состоянии. Информация о том, какой именно блок основной памяти занимает данную строку называется *тегом* (tag) и хранится в связанной с данной строкой ячейке *памяти тегов*. В качестве тега обычно используется часть адреса ОП. Здесь же хранится и информация о состоянии строки. Строка может быть *действительной* (valid), если в ней в текущий момент времени хранится (присутствует) копия соответствующего блока основной памяти, или *недействительной*.

Кэш память

Строка может *достоверно* отражать соответствующий блок основной памяти или быть *модифицированной* (говорят строка «грязная» – dirty). Таким образом, кроме адресной части тега с каждой строкой кэша связаны биты признаков *действительности* (присутствия) V и *модифицированности* M данных. Память тегов представляет собой каталог или справочник кэш-памяти. В операциях обмена с основной памятью строка участвует целиком. Такой кэш называется *несекторированным*. Возможен и вариант *секторированного* кэша, при котором одна строка содержит несколько смежных секторов, размер которых соответствует минимальной порции

Кэш память

При этом в записи каталога, соответствующей каждой строке, должны храниться биты действительности для каждого сектора данной строки. Секторирование позволяет экономить память, необходимую для хранения каталога при увеличении объема кэша, так как при этом увеличивается количество разрядов каждого элемента каталога, а не количество самих элементов (размер каталога).

Кэш память

На эффективность применения кэш-памяти в иерархической системе памяти влияет целый ряд моментов. К наиболее существенным из них можно отнести:

- емкость кэш-памяти;
- размер строки;
- способ отображения основной памяти на кэш-память;
- алгоритм замещения информации в заполненной кэш-памяти;
- алгоритм согласования содержимого основной и кэш-памяти;
- число уровней кэш-памяти.

Кэш память

Реальная эффективность использования кэш-памяти зависит от характера решаемых задач, и невозможно заранее определить, какая ее емкость будет действительно оптимальной. Общая тенденция: по мере увеличения емкости кэш-памяти вероятность промахов сначала существенно снижается, но при достижении определенного значения эффект сглаживается и становится несущественным. Установлено, что для большинства задач близкой к оптимальной является кэш-память емкостью от 1 до 512 Кбайт.

Кэш память

Еще одним важным фактором, влияющим на эффективность использования кэш-памяти, является размер строки. Когда в кэш-память помещается строка, вместе с требуемым словом туда попадают и соседние слова. По мере увеличения размера строки вероятность промахов сначала падает, так как в кэш, согласно принципу локальности, попадает все больше данных, которые понадобятся в ближайшее время. Однако вероятность промахов начинает расти, когда размер строки становится достаточно большим.

Кэш память

Объясняется это тем, что:

• большие размеры строки уменьшают общее количество строк, которые можно загрузить в кэш-память, а малое число строк приводит к необходимости частой их смены;

• по мере увеличения размера строки каждое дополнительное слово оказывается дальше от запрошенного, поэтому такое дополнительное слово менее вероятно понадобится в ближайшем будущем.

Кэш память

Зависимость между размером строки и вероятностью промахов во многом определяется характеристиками конкретной программы, из-за чего трудно рекомендовать определенное значение величины строки. Считается, что наиболее близким к оптимальному является размер строки, равный 4-8 адресуемым единицам (словам или байтам). На практике размер строки обычно выбирают равным ширине шины данных, связывающей кэш-память с основной памятью, или размеру пакета, если процессор поддерживает режим пакетной передачи.

Алгоритмы замещения информации в кэш

Когда кэш-память заполнена, занесение в нее нового блока связано с замещением содержимого одной из строк. При ***прямом*** отображении каждому блоку основной памяти соответствует только одна определенная строка в кэш-памяти, и никакой иной выбор удаляемой строки здесь невозможен. При *полностью* и *частично* ***ассоциативных*** способах отображения требуется какой- либо алгоритм замещения (выбора удаляемой из кэш-памяти строки).

Алгоритмы замещения информации в кэш

Основная цель стратегии замещения – *удерживать* в кэш-памяти строки, к которым наиболее вероятны обращения в ближайшем будущем, и *заменять* строки, доступ к которым произойдет в более отдаленном времени или вообще не случится. Оптимальным будет алгоритм, который замещает ту строку, обращение к которой в будущем произойдет позже, чем к любой другой строке кэша. Такое предсказание практически нереализуемо, поэтому используются алгоритмы, уступающие оптимальному. В любом случае для достижения высокой скорости алгоритм замещения должен

Алгоритмы замещения информации в кэш

Наиболее распространенными являются четыре алгоритма замещения, рассматриваемые в порядке уменьшения их относительной эффективности.

Алгоритм замещения на основе наиболее давнего использования (LRU – Least Recently Used). Является наиболее эффективным алгоритм замещения. В соответствии с этим алгоритмом замещается та строка кэш-памяти, к которой дольше всего не было обращения. Проводившиеся исследования показали, что алгоритм LRU работает достаточно хорошо в сравнении с оптимальным алгоритмом.

Алгоритмы замещения информации в кэш

Наиболее известны два способа аппаратной реализации этого алгоритма.

В ***первом*** из них с каждой строкой кэш-памяти связывается счетчик. К содержимому всех счетчиков через определенные интервалы времени добавляется единица. При обращении к строке ее счетчик обнуляется. Таким образом, наибольшее число будет в счетчике той строки, к которой дольше всего не было обращений, и эта строка – первый кандидат на замещение.

Алгоритмы замещения информации в кэш

Второй способ реализуется с помощью очереди, куда в порядке заполнения строк кэш-памяти заносятся ссылки на эти строки. При каждом обращении к строке ссылка на нее перемещается в конец очереди. В итоге первой в очереди каждый раз оказывается ссылка на строку, к которой дольше всего не было обращений. Именно эта строка, прежде всего и заменяется.

Алгоритмы замещения информации в кэш

Алгоритм, работающий по принципу FIFO (первый вошел, первый вышел – First In First Out). В соответствии с этим алгоритмом заменяется строка, дольше всего находившаяся в кэш-памяти. Алгоритм легко реализуется с помощью рассмотренной очереди, с той лишь разницей, что после обращения к строке положение соответствующей ссылки в очереди не меняется.

Алгоритмы замещения информации в кэш

Произвольный выбор строки для замены.

Простейший алгоритм, в соответствие с которым замещаемая строка выбирается случайным образом. Реализовано это может быть, например, с помощью счетчика, содержимое которого увеличивается на единицу с каждым тактовым импульсом, вне зависимости от того, имело место попадание или промах. Значение в счетчике определяет заменяемую строку.

Данный алгоритм используется крайне редко.

Алгоритмы замещения информации в кэш

Алгоритм замены наименее часто использовавшейся строки (LFU – Least Frequently Used). В соответствии с этим алгоритмом заменяется та строка в кэш-памяти, к которой было меньше всего обращений.

Аппаратная реализация алгоритма: каждая строка связывается со счетчиком обращений, к содержимому которого после каждого обращения добавляется единица. Главным претендентом на замещение является строка, счетчик которой содержит наименьшее число.

Произвольный выбор строки для замены.

Простейший алгоритм, в соответствие с которым

Кэш память

Одной из проблем является отсутствие баланса между задержками кэша и интенсивностью попаданий. Большие кэши имеют более высокий процент попаданий, но и большую задержку. Чтобы ослабить противоречие между этими двумя параметрами, большинство компьютеров использует несколько уровней кэша, когда после маленьких и быстрых кэшей находятся более медленные большие кэши (в настоящий момент — до 3 уровней в иерархии кэшей L1,L2,L3).

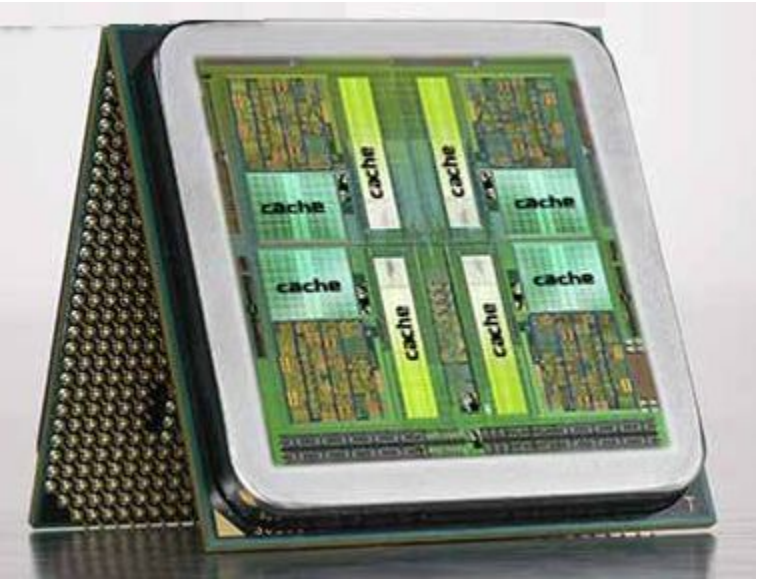
Многоуровневые кэши обычно работают в последовательности от меньших кэшей к большим. Сначала происходит проверка наименьшего и наибыстрейшего кэша первого уровня, в случае попадания процессор продолжает работу на высокой скорости. Если меньший кэш дал промах, проверяется следующий, чуть больший и более медленный кэш второго уровня, и так

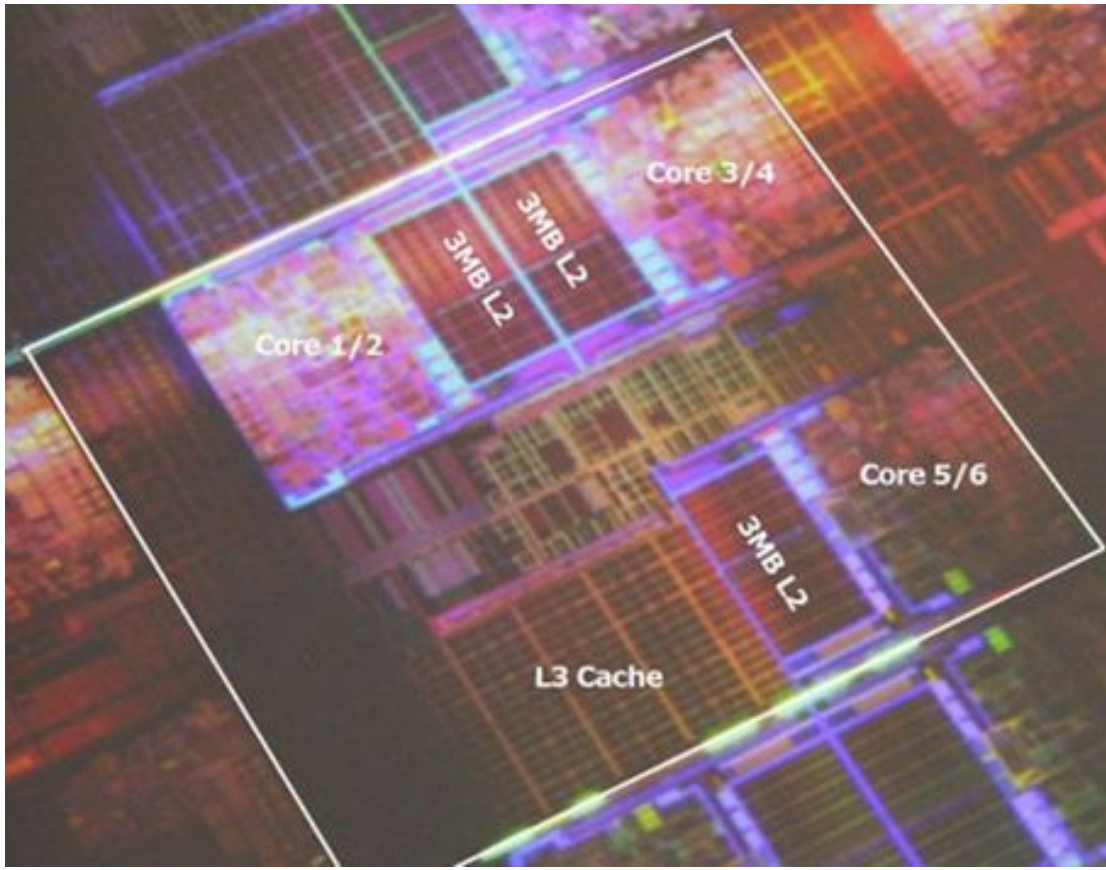
Кэш память


2
Для большинства запросов в кэш-памяти есть данные, их перехватывает контроллер
КЭШ-ПАМЯТЬ
ПРОЦЕССОР
КЭШ-КОНТРОЛЛЕР
ОПЕРАТИВНАЯ ПАМЯТЬ
1
Процессор генерирует большое количество запросов к Опертивной памяти
3
Лишь небольшая часть запросов требует обращения к Оперативной памяти

Кэш память

Кэш память 1-го уровня расположена в ядре.





Иерархия кэшпамяти в микроархитектуре Intel Core

Рассмотрим пример организации многоуровневой кэш памяти в процессорах с микроархитектурой Intel Core.
Все современные процессоры Intel основаны на базе микроархитектуры Intel Core (которая в плане кэша не отличается от микроархитектуры Nehalem) и имеют трехуровневую подсистему кэшпамяти.
Кэшпамять первого уровня L1 делится на 8-канальный 32-килобайтный кэш данных (L1D) и 4-канальный 32-килобайтный кэш инструкций (L1I).
Каждое ядро процессора имеет унифицированный (единый для инструкций и данных) кэш второго уровня (L2) размером 256 Кбайт. Кэш L2 является также 8-канальным, а размер строки кэша составляет 64 байт.

Иерархия кэшпамяти в микроархитектуре Intel Core

Кэш третьего уровня (L3) разделяется между всеми ядрами процессора. Его размер зависит от количества ядер процессора. Кэш L3 является 16-канальным.

Кэш L3 — включающий по своей архитектуре по отношению к кэшам L1 и L2, то есть в кэше L3 всегда дублируется содержимое кэшей L1 и L2. А вот кэши L1 и L2 не являются ни включающими, ни исключающими по отношению друг к другу, то есть кэш L2 может содержать, а может и не содержать копию данных кэша L1.

Применение именно включающего кэша L3 имеет свои преимущества по сравнению с исключающей архитектурой.

Иерархия кэшпамяти в микроархитектуре Intel Core

Рассмотрим несколько характерных примеров чтения данных из кэша L3 в четырехъядерном процессоре Intel. Предположим сначала, что ядро процессора Core 0, обнаружив, что требуемых ему данных нет ни в кэше L1, ни в кэше L2, обращается к кэшу L3. Если необходимых данных нет также и в кэше L3, то в случае исключающей архитектуры кэша L3 потребовалось бы проверить и наличие требуемых данных в кэшах L1 и L2 каждого из ядер Core 1, Core 2 и Core 3. При включающей архитектуре кэша L3 надобность в подобной проверке отпадает, поскольку включающая архитектура кэша L3 гарантирует, что при отсутствии данных в кэше L3 их также не будет в кэшах L1 и L2.

Иерархия кэшпамяти в микроархитектуре Intel Core

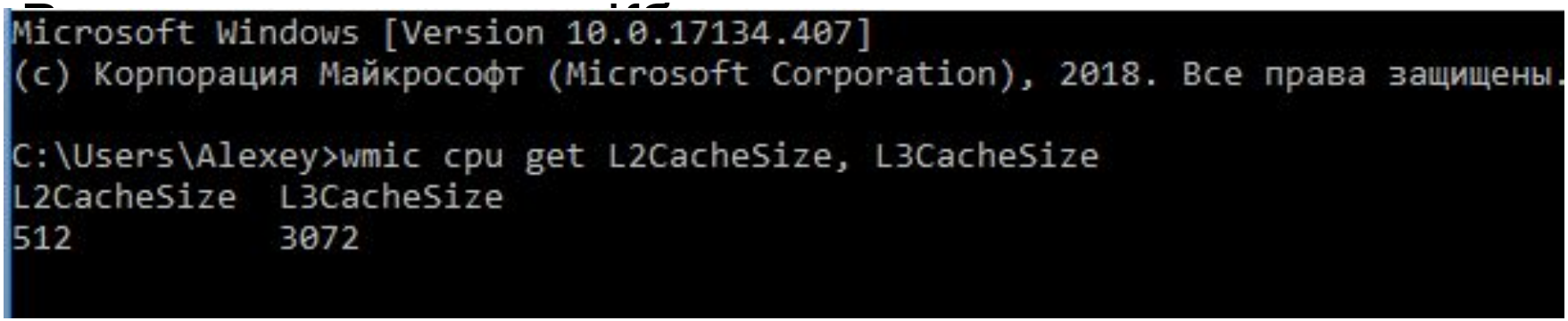
Если же требуемые ядру Core 0 данные обнаруживаются в кэше L3, то при исключающей архитектуре кэша больше не нужно выполнять никаких действий, поскольку данная архитектура гарантирует их отсутствие в кэшах L1 и L2 ядер Core 1, Core 2 и Core 3. Однако при включающей архитектуре кэша L3 наличие требуемых данных в кэше L3 означает, что они также содержатся в какомнибудь из кэшей ядер Core 1, Core 2 или Core 3. Но в этом случае не нужна дополнительная проверка кэшей L1 и L2 всех остальных ядер. Достигается это тем, что в теге кэшстроки L3-кэша записывается, к какому из ядер принадлежат данные, поэтому достаточно лишь прочитать содержимое этого тега.

Тестирование размера кэша

- Если кэшпамять первого уровня L1 имеет 64Кб на ядро, то L2 и L3 могут колебаться. Для их поиска открываем командную строку через меню «Пуск» (вводим значение «cmd» через строку поиска) или комбинацией клавиш «**Win»+»R**»..

Далее в командной строке необходимо записать

«wmic cpu get L2CacheSize, L3CacheSize»



Кэш 2-го уровня – 512 Кб (256 Кб на ядро)

Кэш 3-го уровня – 3072 Кб

Постоянное запоминающее устройство

Постоянное запоминающее устройство (**ПЗУ, ROM - Read Only Memory**) — энергонезависимая память, используется для хранения массива неизменяемых данных. Постоянные запоминающие устройства бывают нескольких видов.

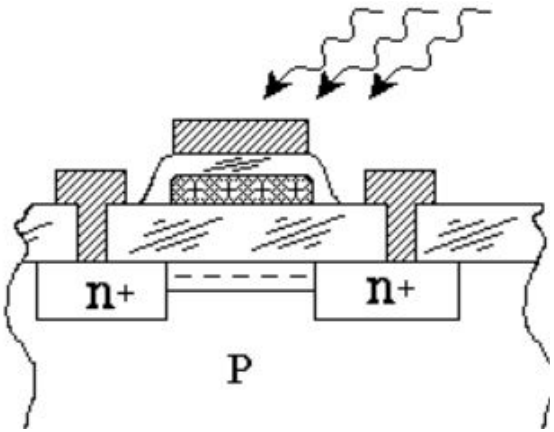
Непрограммируемые (масочные) ПЗУ (MROM - Masked Read Only Memory). В масочных ПЗУ элементами памяти в накопителе могут быть диоды, биполярные транзисторы, МОП транзисторы. Программирование масочных ПЗУ выполняется однократно в процессе изготовления. Оно заключается в нанесении при помощи фотошаблонов (масок) в нужных потребителю местах контактных соединений (в виде диодов или транзисторов). Занесенную в масочное ПЗУ информацию в технической документации называют «прошивкой». Перепрограммировать масочное ПЗУ невозможно.

Программируемые ПЗУ

Программируемые ПЗУ (PROM - Programmable Read Only Memory) предоставляют пользователю возможность самостоятельно его запрограммировать. Содержимое PROM формируется после того, как устройство изготовлено. Процесс занесения информации в PROM осуществляется специальным устройством, которое называется программатором. Обычно этот процесс основан на пережигании плавких перемычек. Этот процесс необратим, т.е. после такого программирования содержимое памяти не может быть изменено. PROM - получило довольно широкое распространение.

Стираемые программируемые ПЗУ

Стираемые программируемые (репрограммируемые) ПЗУ (EPROM - Erasable Programmable Read Only Memory) обеспечивают возможность неоднократного изменения своего содержимого. Это производится путем стирания с помощью интенсивного ультрафиолетового излучения. Стирание осуществляется за 10-15 минут. **ПЗУ с ультрафиолетовым стиранием** строится на основе запоминающей матрицы построенной на ячейках памяти, внутреннее устройство которой приведено на следующем



Ячейка представляет собой МОП транзистор, в котором затвор выполняется из поликристаллического кремния. Затем в процессе изготовления микросхемы этот затвор окисляется и в результате он будет окружен оксидом кремния — диэлектриком с прекрасными изолирующими свойствами.

Стираемые программируемые ПЗУ (структура)

При полностью стертом ПЗУ, заряда в плавающем затворе нет, и поэтому транзистор ток не проводит. При программировании ПЗУ, на второй затвор, находящийся над плавающим затвором, подаётся высокое напряжение и в плавающий затвор за счет туннельного эффекта индуцируются заряды. После снятия программирующего напряжения индуцированный заряд остаётся на плавающем затворе, и, следовательно, транзистор остаётся в проводящем состоянии. Заряд на плавающем затворе подобной ячейки может храниться десятки лет. Количество циклов записи-стирания микросхем EPROM находится в диапазоне от 10 до 100 раз, после чего микросхема РПЗУ выходит из строя. Это связано с разрушающим

олетового излучения на оксид кремния.

Для того, чтобы этот свет мог беспрепятственно проходить к полупроводниковому кристаллу, в корпус микросхемы ПЗУ встраивается окошко из кварцевого стекла.

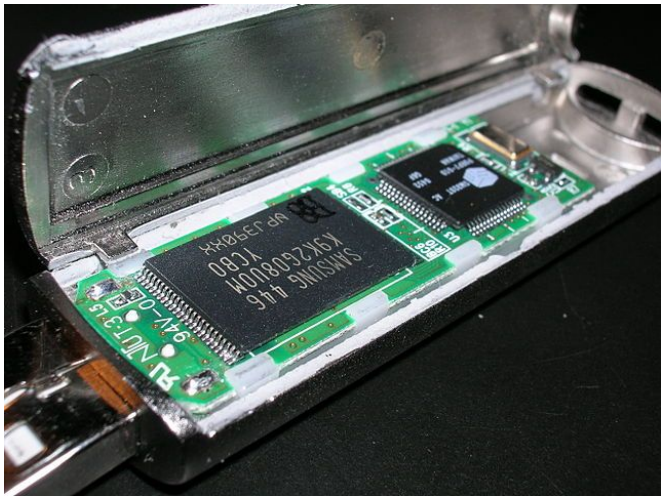
Электрически стираемые программируемые ПЗУ

Электрически стираемые программируемые ПЗУ (EEPROM - Electrically Erasable Read Only Memory). Такое устройство предоставляет возможность стирать свое содержимое не ультрафиолетовыми лучами, а электрическими сигналами высокого уровня (~20В). В качестве запоминающей ячейки в них используются ячейки аналогичные EPROM, но они стираются электрическим потенциалом, поэтому количество циклов записи - стирания для этих микросхем достигает 1000000 и более. Время стирания ячейки памяти в таких микросхемах значительно уменьшилось. Тем не менее время стирания существенно больше времени считывания, поэтому такие ППЗУ записывают данные значительно медленнее чем считывают.

EEPROM дороже и меньше по объему, но зато позволяют перезаписывать каждую ячейку памяти отдельно. Область применения электрически стираемых ПЗУ - хранение данных, которые не должны стираться при выключении питания. Запись в такую память возможна в самом компьютере в специальном режиме работы.

FLASH - ПЗУ

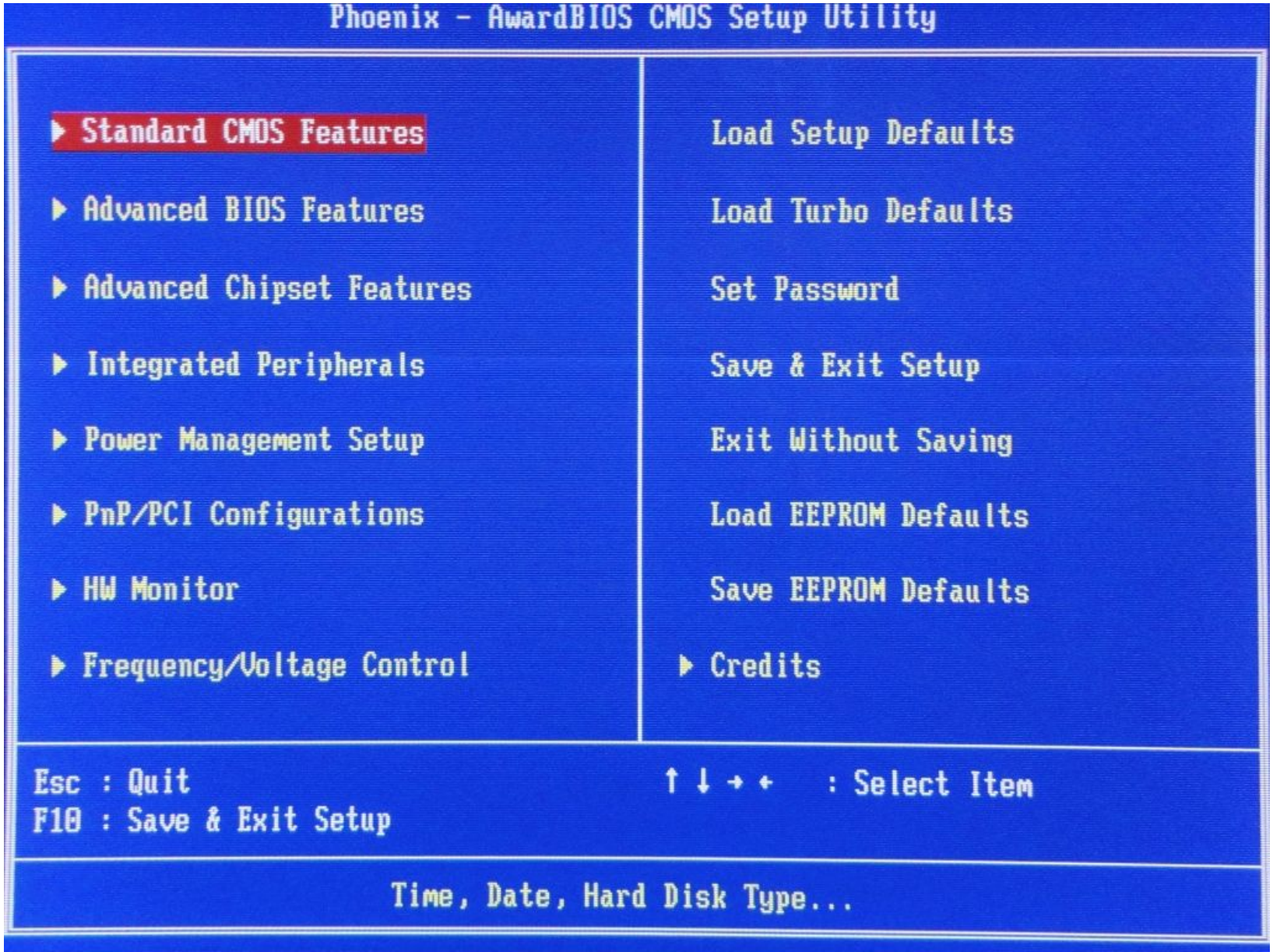
- FLASH - ПЗУ отличаются от ЭСППЗУ тем, что стирание производится не каждой ячейки отдельно, а всей микросхемы в целом или блока запоминающей матрицы этой микросхемы, как это делалось в РПЗУ .



BIOS

Для того, что бы процессор смог начать выполнять команды операционной системы, они должны находиться в ОЗУ. Но во время включения ПК оперативная память пуста, так как является энергозависимой. Используется специальную небольшую программу, получившую название BIOS, иногда называемую начальным загрузчиком. Термин BIOS (Basic Input/Output System) - - «Базовая система ввода/вывода». Такое название получил набор микропрограмм, отвечающих за работу базовых функций видеоадаптеров, дисплеев, дисковых накопителей, дисководов, клавиатур, мышей и других основных устройств ввода/вывода информации.

BIOS


Phoenix - AwardBIOS CMOS Setup Utility
Standard CMOS Features
Advanced BIOS Features
Advanced Chipset Features
Integrated Peripherals
Power Management Setup
PnP/PCI Configurations
HW Monitor
Frequency/Voltage Control
Load Setup Defaults
Load Turbo Defaults
Set Password
Save & Exit Setup
Exit Without Saving
Load EEPROM Defaults
Save EEPROM Defaults
Credits
Esc : Quit
F10 : Save & Exit Setup
↑ ↓ → ← : Select Item
Time, Date, Hard Disk Type...

Функции BIOS

На ROM BIOS возложено выполнение 3-х основных функций:

- предоставляет операционной системе аппаратные драйверы и осуществляет сопряжение между материнской платой и остальными устройствами персонального компьютера. ROM BIOS должен быть настроен на особенности конкретной материнской платы.
- содержит тест проверки системы, так называемый POST (Power On Self Test), который проверяет при включении персонального компьютера все важнейшие компоненты.
- содержит программу установки параметров BIOS и аппаратной конфигурации персонального компьютера - CMOS Setup.

С учетом того, что BIOS отвечает за самый начальный этап загрузки компьютера, то эта программа должна быть доступна для базовых устройств сразу же после нажатия на кнопку включения ПК. Именно поэтому она хранится не на жестком диске, как большинство обычных приложений, а записывается в специальную микросхему флэш-памяти, расположенную на системной плате.

История BIOS

В самых первых компьютерах для хранения BIOS использовались микросхемы постоянной памяти (ПЗУ или ROM), запись на которые самого кода программы единожды осуществлялась на заводе. Несколько позже стали использовать микросхемы EPROM и EEROM, в которых имелась возможность в случае необходимости осуществлять перезапись BIOS, но только с помощью специального оборудования. В современных же персональных компьютерах BIOS хранится в микросхемах, созданных на основе флэш-памяти, перезаписывать которые можно с помощью специальных программ прямо на ПК в домашних условиях. Такая процедура обычно называется перепрошивкой и требуется для обновления микропрограммы до новых версий или ее замены в случае повреждения.

Интерфейс для начальной загрузки ПК UEFI

Интерфейс для начальной загрузки ПК UEFI (Unified Extensible Firmware Interface). В 2011 году, с запуском в производство материнских плат для процессоров Intel поколения Sandy Bridge началось массовое внедрение нового программного интерфейса для начальной загрузки компьютера - UEFI. В отличие от BIOS, код UEFI и вся ее служебная информация может храниться не только в специальной микросхеме, но и на разделах как внутренних, так и внешних жестких дисков, а так же сетевых хранилищах. В свою очередь, тот факт, что загрузочные данные могут размещаться на вместительных накопителях, позволяет за счет модульной архитектуры наделять UEFI богатыми функциональными возможностями. Например, это могут быть развитые средства диагностики, или полезные утилиты, которые можно будет использовать как на этапе начальной загрузки ПК, так и после запуска ОС.

Интерфейс для начальной загрузки ПК UEFI

При загрузке этой системы, которую, многие называют своеобразной мини-ОС, обращает внимание на себя факт поддержки мыши и возможность установки для интерфейса регионального языка. Можно заметить, что, в отличие от BIOS, UEFI может работать с поддержкой сетевых устройств и отображать оптимальные режимы работы некоторых компонентов установленного оборудования.

Преимущества UEFI

Поддержка UEFI имеет и ряд преимуществ, среди которых можно выделить следующие:

- простой интуитивно понятный интерфейс;
- поддержка региональных языков и управления мышью; работа с дисками 2 Тб и выше;
- более быстрая загрузка операционной системы;
- наличие собственного загрузчика;
- возможность работы на базе процессоров с архитектурой x86, x64 и ARM;
- возможность подключения к локальным и виртуальным сетям с доступом в интернет;
- наличие собственной системы защиты от проникновения вредоносных кодов и вирусов;
- упрощенное обновление.

Необходимость иерархии компьютерных памятей

Практически с самого начала развития электронной вычислительной техники имелась проблема «бесперебойного» питания центрального процессора обрабатываемыми данными. Изготовление ёмкой памяти на тех же элементах, что и регистры вычислительных блоков процессора, было дорого и не решало задачу долговременного хранения больших объёмов информации, поскольку при отключении питания она стирается. Устройства же долговременного, энергонезависимого хранения данных медлительны. Однако замечено, что большинство алгоритмов обращаются в каждый промежуток времени к сравнительно небольшому набору данных, который может быть помещен в небольшую, но быструю (хотя и дорогую) память. Иерархичность организации памяти компьютера означает, что различные виды памяти образуют «пирамиду», при движении по которой вниз увеличивается объём, падает стоимость хранения единицы информации, но одновременно увеличивается время доступа к данным. На более высоких уровнях, нижние уровни иерархической организации могут подготавливать данные укрупненными частями с буферизацией и, по наполнению буфера, сообщать верхнему уровню о готовности передачи данных

Иерархия компьютерных памятей

В большинстве современных ПК используется следующая иерархия памяти:

- регистровая память процессора — наиболее быстрый доступ (порядка 1 машинного такта), но размером лишь в несколько десятков-сотен байт;
- кэш процессора 1-го уровня (Cache L1) — время доступа порядка нескольких тактов, размером в десятки килобайт;
- кэш процессора 2-го уровня (Cache L2) — большее время доступа (от 2 до 10 раз медленнее L1), обычно несколько сотен килобайт;
- кэш процессора 3-го уровня (Cache L3) — время доступа около сотни тактов, размер в несколько мегабайт;
- оперативная память (ОП) — время доступа порядка единиц нс, размеры - единицы и десятки Гбайт. Отметим, что кэш-памяти современных ПК размещаются на одном кристалле с микропроцессором и работают на одинаковой с ним тактовой частоте (2 – 4 ГГц), которая значительно превосходит частоту обращения к ОП, размещаемой на материнской плате (сотни МГц);
- дисковые накопители — с временем доступа единицы-

Схема иерархии компьютерных памятей